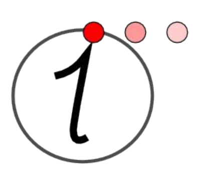
UMLMON

Bedienungsanleitung

FÜR VERSION 1.0.4



Informatikbüro
Dipl.-Inform. Gerd Stolpmann

Viktoriastr. 45 • 64293 Darmstadt
http://www.gerd-stolpmann.de

Informatikbüro Dipl.-Inform. Gerd Stolpmann

Inhaltsverzeichnis

1 Einleitung

UMLMON ist kein ganz einfaches Produkt. Daher haben wir diese ausführliche Anleitung geschrieben. Natürlich wird immer noch eine ganze Menge Fachwissen benötigt, vor allem über die System-Administration von Linux und über Netzwerk-Administration. Dies lässt sich leider nicht vermeiden, weil es in der Natur der Sache liegt.

UMLMON läuft auf aktuellen x86-Linux-Systemen, die wenigstens mit glibc-2.3 ausgestattet sind. In dieser Anleitung gehen wir näher auf Debian und SUSE Linux ein. Das hat einfach damit zu tun, dass der Autor sich in beiden Systemen gut auskennt und weiterführende Tipps geben kann.

Debian: Zur Zeit ist die aktuelle Version 3.1 (auch als „Sarge" bekannt). UMLMON funktioniert gut auf Debian Sarge. Die Vorgängerversion 3.0, Codename Woody, wird nicht unterstützt. Die Abspaltung **Ubuntu** wird unterstützt.

SUSE: UMLMON wurde auf SUSE Linux Professional 9.2 getestet. Es ist unklar, ob es auf älteren Versionen funktioniert.

Eine der Komplikationen dieser Anleitung ist, dass stets zwei Programme im Auge behalten werden müssen, nämlich:

1. Eben UMLMON, den Monitor
2. Die eigentliche virtuelle Maschine, die aus einem laufenden User Mode Linux (UML)-Kernel besteht, und die von UMLMON gesteuert wird.

Jedes der beiden Programme kann aktiv oder inaktiv sein. Da die Aufgabe von UMLMON die Steuerung des ganzen ist, wird in diesem Modell zuerst der Monitor aktiviert, und der Monitor bekommt dann die Anweisung, den UML-Kernel zu starten (oder herunterzufahren). Sie können mittels UMLMON auch zahlreiche weitere Anweisungen geben; manche werden an den UML-Kernel weitergeleitet, manche werden von UMLMON selbst ausgeführt.

Sie können natürlich auch den UML-Kernel ohne UMLMON starten; es sind eben zwei unabhängige Programme. Sie verlieren damit allerdings zahlreiche administrative Möglichkeiten.

Wichtiger Hinweis: Das Informatikbüro Stolpmann hat nur UMLMON hergestellt, nicht aber den UML-Kernel. Aus diesem Grunde kann auch nur die Verantwortung für UMLMON übernommen werden. Wenn also der UML-Kernel nicht funktioniert wie er soll, ist dies, kurz gesagt, Ihr eigenes Problem. Da der UML-Kernel freie Software ist, können Sie sich dieses Geschenkes jederzeit erfreuen, ihn von zahlreichen Servern herunterladen und selber erproben. Sie können dies ganz unabhängig tun von der Existenz von UMLMON. Wenn Sie UMLMON einsetzen, gehen wir davon aus, dass Sie sich selbst von der Qualität des UML-Kernels überzeugt haben und der Meinung sind, dass er für Ihre Einsatzzwecke geeignet ist, und nun eine professionelle Betriebsumgebung aufsetzen möchten.

Wir liefern übrigens auch keine UML-Kernel aus. In Anhang A haben wir zusammengefasst, wie man UML-Kernel selbst herstellt.

Etwas ähnliches gilt für das Betriebssystem, das Sie in der virtuellen Maschine ablaufen lassen. Für Fehler in Debian und SUSE sind eben Debian und SUSE verantwortlich. In Anhang B haben wir beschrieben, wie man diese Systeme in die virtuellen Maschinen hineininstalliert.

Diese Anhänge dienen nur als Hilfestellung und geben keine Gewährleistung, dass mit den beschriebenen Methoden stabile Systeme aufgesetzt werden können.

2 Planung des Einsatzes von UMLMON

In diesem Abschnitt werden die Planungsentscheidungen durchgegangen, die bei einem Einsatz von UMLMON anstehen. Da UMLMON eine komplexe Betriebsumgebung zur Verfügung stellt, ist ein gewisses Maß an Planung unverzichtbar.

2.1 Auswahl von Linux-Kerneln

Die erste wichtige Frage lautet: Welche Linux-Kernel-Versionen sollen auf dem Host- und dem Gast-System zum Einsatz kommen?

Als Host-Kernel kann jeder unveränderter Distributions-Kernel der Kernel-Serie 2.6 verwendet werden, allerdings um den Preis, dass der performanteste SKAS3-Modus nicht möglich ist. User Mode Linux kann in drei Modi betrieben werden, die als TT-Modus (für *tracing thread*), SKAS0- und SKAS3-Modus (für *separate kernel address space*) bekannt sind. Der TT-Modus benutzt nur Linux-Features, die in allen Host-Kerneln vorhanden sind, ist aber relativ langsam und hat Sicherheitsmängel. Es wird daher dringend vom produktiven Einsatz des TT-Modus abgeraten. Der TT-Modus wird zudem in künftigen UML-Kerneln vermutlich nicht mehr unterstützt. Der SKAS-Modus behebt diese Probleme. Es gibt ihn in zwei Varianten: SKAS0 und SKAS3. SKAS0 läuft auf ungepatchten Linux-Kerneln (die „0" im Namen soll vermutlich versinnbildlichen, dass 0 Patches benötigt werden), ist aber geringfügig langsamer als SKAS3. Letzterer ist am schnellsten, setzt aber einen gepatchten Host-Kernel voraus. Um den Patch einzuspielen, müssen Sie (1) den Linux-Quellcode besorgen, (2) den Patch auf den Quellcode anwenden und (3) Linux neu übersetzen. Zu guter Letzt muss der neue Linux-Kernel auf dem Host-System aktiviert werden. In Anhang A haben wir diese Schritte weiter erläutert.

Im Gast-Kernel müssen die User Mode Linux-Features aktiviert sein. Seit Kernel 2.6.9 ist UML offizieller Teil des Linux-Quellcodes, so dass es ausreicht, die Kernel-Quellen mit bestimmten UML-spezifischen Konfigurationsoptionen neu übersetzen. Das Ergebnis der Kernel-Übersetzung ist dann ein Executable *linux*, das Sie auf der Kommandozeile wie ein Programm aufrufen können. Das bedeutet, dass Sie für den Gast-Kernel folgende Schritte einplanen müssen: (1) Linux-Quellcode besorgen, (2) eine UML-fähige Kernel-Konfiguration einrichten und (3) Linux

übersetzen. Das so erhaltene *linux*-Executable müssen Sie dann nur noch in das richtige Verzeichnis kopieren. (Siehe wiederum Anhang A für Details.)

Hinweis: In früheren Ausgaben dieses Handbuchs haben wir auch dargestellt, wie man UML für die ältere Linux-Reihe 2.4 bereitstellt. Da diese kaum noch eingesetzt wird, haben wir entsprechende Passagen gelöscht. Es ist auch davon auszugehen, dass die Unterstützung für Linux 2.4 durch mangelnde Wartung inzwischen fehlerhaft ist.

Hinweis: Ältere Linux-Versionen bis 2.6.16 unterstützen unter UML nicht NPTL, die neue Multi-Threading-Library für 2.6-Kernel. Wenn diese Versionen als Gast-Kernel eingesetzt werden, muss NPTL daher in den Gast-Systemen deaktiviert werden. Seit Linux-2.6.17 ist dieses Problem jedoch behoben.

2.2 Planung des Host-Hauptspeichers und der CPU

Das Host-System muss mit viel Hauptspeicher ausgestattet werden! Generell sollten Sie *mindestens* so viel Hauptspeicher haben wie die Summe des Hauptspeichers, der für die virtuellen Maschinen konfiguriert wird. Es ist empfehlenswert, über diesen errechneten Wert deutlich hinauszugehen, da das Host-System zur Abwicklung der I/O-Operationen viel Pufferplatz benötigt.

Sie sollten pro virtueller Maschine nicht mehr als 1,75 GB Hauptspeicher konfigurieren.

User Mode Linux läuft auch auf SMP-Systemen, und die CPU-Last, die die virtuellen Maschinen erzeugen, verteilt sich in diesem Fall automatisch über die vorhandenen CPUs. In der Grundeinstellung wird jede virtuelle Maschine aber nur eine CPU saturieren können. Mit Spezialeinstellungen (die allerdings als experimentell gelten) können die virtuellen Maschinen selbst als SMP-Systeme konfiguriert werden, d.h. jede Maschine kann mehrere virtuelle CPUs simulieren (unabhängig von der tatsächlichen CPU-Zahl). Die Performance-Auswirkungen von virtuellem SMP sind jedoch schwer abzuschätzen.

2.3 Planung der Plattenkonfiguration

Die virtuellen Maschinen benötigen viel Plattenplatz. Aus diesem Grund ist es unverzichtbar, rechtzeitig zu planen, wie viel Platz benötigt wird, und auf welchem Dateisystem dieser zur Verfügung gestellt wird. Die virtuellen Maschinen haben jeweils virtuelle Festplatten, die durch Dateien dargestellt werden. Es gibt dabei zwei Möglichkeiten: Flache Images und so genannte Copy-On-Write-Images. Eine virtuelle Festplatte, die als flaches Image dargestellt wird, enthält 1:1 die Daten des Images (d.h. ein Block der virtuellen Festplatte entspricht einem Block der Datei). Flache Images sind einfach und die Grundform der virtuellen Festplatte. Ein Copy-On-Write- oder COW-Image versucht, Plattenplatz zu sparen, wenn mehrere sehr ähnliche virtuelle Festplatten vorhanden sind, indem nur die Unterschiede gespeichert werden.

Wir gehen hier davon aus, dass Sie N > 1 virtuelle Maschinen installieren wollen. Es kommt nun darauf an, ob diese Maschinen ähnlich aufgesetzt werden sollen (z.B. mit der gleichen Software-Ausstattung) oder ob jede Maschine verschieden ist.

Wenn die Maschinen alle verschieden sind, ist die Rechnung einfach, aber hoch: Sie benötigen für jede Maschine so viel Platz wie für eine eigenständige Linux-Installation auf einem echten Rechner. Sie haben nun die Wahl, ob Sie alle virtuellen Festplatten auf dasselbe Dateisystem ablegen wollen oder diese verteilen wollen. Letzteres kann die Performanz von Plattenoperationen stark erhöhen, da parallele Zugriffe auf mehrere virtuelle Festplatte beschleunigt werden. Ob dies für Sie wichtig ist, muss im Einzelfall geklärt werden.

> **Beispiel:** Sie wollen drei virtuelle Maschinen aufbauen, die drei verschiedenen Linux-Distributionen entsprechen. Sie kalkulieren, dass Sie jeweils 10 GB benötigen. In der Summe brauchen Sie also 30 GB Plattenplatz. Da Sie die Maschinen nicht parallel benutzen, haben Sie nichts davon, die virtuellen Festplatten auf mehrere Dateisysteme zu verteilen. Daher kaufen Sie eine echte Festplatte mit 30GB Platz und bringen alle virtuellen Festplatten auf ihr unter.

Wenn die Maschinen alle gleichartig sind, können Sie viel Platz sparen, indem Sie den COW-Mechanismus in Anspruch nehmen. Hierfür müssen Sie eine Vorlagen-Maschine erzeugen, die selbst nicht benutzt wird, aber deren Festplatten den Referenzinhalt vorgeben. Die eigentlichen virtuellen Maschinen, die benutzt werden, speichern dann nur noch die Unterschiede zu der Referenz. Es empfiehlt sich, den COW-Mechanismus nur für solche Dateibäume zu verwenden, die sich selten ändern, beispielsweise die Dateibäume, die das Betriebssystem enthalten. Diese Dateibäume werden auf lange Zeit auf allen virtuellen Maschinen ähnlich bleiben und eignen sich daher für COW.

Dies bedeutet, dass Sie entscheiden müssen, welche Dateibäume für COW in Frage kommen. Diese Dateibäume sollten in separaten virtuellen Festplatten abgelegt sein, für die der COW-Mechanismus aktiviert wird. Die anderen Dateibäume werden durch normale flache Images dargestellt.

Die Rechnung ist dann wie folgt: Sie benötigen Platz für die Referenz, auf die sich die COW-Platten beziehen. Für die COW-Platten selbst benötigen Sie nur einen Bruchteil der Plattengröße als Platz. Hierzu sollten Sie überlegen, wie viel Prozent der Dateien über die gesamte Lebenszeit ausgetauscht oder überschrieben werden. Diese Größe ist der Platz, den Sie für die COW-Platten benötigen.

> **Beispiel:** Sie haben 5 Maschinen, die mit der selben Linux-Distribution und der selben Software-Ausstattung versehen sind. Als COW-geeigneten Dateibaum stufen Sie /usr ein. Ungeeignet sind /var, /tmp und /data. Die anderen Verzeichnisse unterhalb / enthalten nur wenige MB und können in der Rechnung ignoriert werden. Pro Maschine haben Sie somit fünf Festplatten:
> /, /usr, /var, /tmp und /data. Für /usr richten Sie COW ein, die anderen Platten

werden als flache Images repräsentiert.

Die Referenz für /usr benötigt 3 GB. Sie erwarten, dass 10 % des Dateivolumens über die Lebenszeit ausgetauscht wird (z.B. durch Einspielen von Patches). Damit erwarten Sie einen Platzbedarf von 3 GB + 5 × (10 % von 3 GB) = 4,5 GB.

Für die anderen vier virtuellen Platten errechnen Sie einen Gesamtbedarf von 3 GB pro Maschine. Zusammen ergibt sich ein Bedarf von 4,5 GB + 5 × 3 GB = 19,5 GB. Hätten Sie auf COW verzichtet, läge der Bedarf bei 5 × 6 GB = 30 GB. Sie haben also ungefähr die Hälfte gespart.

Es bleibt noch die Entscheidung, auf welchen Host-Dateisystemen die virtuellen Festplatten untergebracht werden. Dies hängt wiederum davon ab, bei welchen Platten Sie parallele Nutzung erwarten und wie hoch Ihr Performanz-Bedarf ist. Es kann hier nur eine vage Empfehlung ausgesprochen werden: Ermitteln Sie, ob es virtuelle Platten gibt, die stark parallel benutzt werden, und speichern Sie die Image-Dateien auf verschiedenen echten Platten. Damit vermeiden Sie, dass die virtuellen Platten um die selbe echte Platte konkurrieren.

2.4 Planung der Benutzerkonfiguration

Die virtuellen Maschinen werden bei UMLMON Benutzerkonten zugeordnet. Es wird dringend empfohlen, für diesen Zweck unprivilegierte Konten einzurichten (d.h. nicht root zu verwenden). Ansonsten ergeben sich Sicherheitsmängel!

Sie haben nun die freie Wahl, welche Konten Sie verwenden möchten. Sie können z.B. für alle virtuelle Maschinen dasselbe Konto benutzen. Sie können aber auch für jede Maschine ein eigenes Konto einrichten.

Wenn für mehrere Maschinen dasselbe Konto benutzt wird, ist dies nicht ganz frei von Sicherheitsimplikationen. Generell sind die virtuellen Maschinen zwar vollständig von einander isoliert, d.h. eine Maschine kann nicht auf eine andere zugreifen, und zwar selbst in dem Fall, wenn beide Maschinen demselben Konto zugeordnet sind. Es gibt jedoch eine kleine Unsicherheit, was den Zugriff auf die TAP-Netzwerk-Interfaces betrifft. Diese sind ebenfalls Konten zugeordnet, und eine virtuelle Maschine kann dann das Interface einer anderen Maschine manipulieren. Falls dies für Sie sicherheitsrelevant ist, sollten Sie die Maschinen keinen gemeinsamen Konten zuordnen.

Ein sehr wichtiger Aspekt bei der Planung der Benutzerkonfiguration ist die Frage, wie Sie den administrativen Zugriff auf die Maschinen regeln wollen. Im Prinzip kommen hier zwei Möglichkeiten in Betracht:

1. Zur Administration einer Maschine muss man sich auf das Host-System einloggen, und zwar für das Konto, dem die Maschine zugeordnet ist. Zur Verwendung von umlwatch, der Administrations-Shell, ist in diesem Fall kein

weiteres Passwort vorgesehen. Außerdem können alle Dateien, die zu der Maschine gehören, direkt manipuliert werden.

2. Zur Administration einer Maschine muss man sich auf ein eigenes Shell-System einloggen (dies kann ein als Kommandozeilensystem konfiguriertes Linux-System oder die web-basierte Oberfläche UMLMON-Web sein). Der administrative Zugriff erfolgt nun über die RPC-Netzwerk-Schnittstelle von UMLMON. In diesem Fall besitzt jede Maschine ein Passwort, das zur Administration eingegeben werden muss. Ein direkter Zugriff auf die Dateien der Maschine ist nicht möglich.

Wenn Sie das Host-System vor unberechtigtem Zugriff stark schützen möchten, raten wir von der ersten Möglichkeit ab. Wenn man sich auf ein System direkt einloggen kann, ist es schwer zu übersehen, welche Rechte die einzelnen Benutzer tatsächlich haben. Es gibt allerdings die Möglichkeit, den ssh-Zugriff auf den Aufruf der umlwatch-Shell zu beschränken (siehe Abschnitt 5.3); möglicherweise reicht dieser Sicherheitsstandard für Sie aus. Für die zweite Variante spricht außerdem, dass die Benutzerkonten, für die die virtuellen Maschinen konfiguriert sind, nicht übereinstimmen müssen mit den Benutzerkonten, unter denen sich die Administratoren der Maschinen am Shell-System anmelden. Nachteilig ist der höhere Aufwand zur Einrichtung des Shell-Systems.

2.5 Optional: Planung der Netzwerkkonfiguration

Wenn Sie keinen Netzwerkzugriff benötigen, können Sie diesen Punkt überspringen. In der Regel wird dies aber der Fall sein.

User Mode Linux erlaubt zahlreiche Netzwerkkonfigurationen. UMLMON kann im Prinzip mit allen Konfigurationsoptionen betrieben werden. Für eine Option, die Host-to-Host-Anbindung, enthält UMLMON eine Setup-Funktion, die sie besonders einfach in der Benutzung macht. In dieser Anleitung stellen wir neben der Host-to-Host-Anbindung auch die Konfiguration virtueller Netzwerk-Segmente vor. Zusammen decken diese beiden Möglichkeiten praktisch alle Bedürfnisse ab.

Generell kann die Netzwerkkonfiguration nur durch den Host-Administrator erfolgen, da diese Konfiguration immer Auswirkungen auf die Integrität des Netzwerks hat. Desweiteren müssen ggf. Firewall-Regeln angepasst werden (in jedem Fall, wenn iptables auf dem Host-System verwendet wird, sowie dann, wenn eine externe Firewall den Zugang zu fremden Netzen regelt).

Bei der **Host-to-Host-Anbindung** bekommt die virtuelle Maschine ein virtuelles Netzwerk-Interface, das zwei Seiten hat. Aus Sicht der virtuellen Maschine handelt es sich um ein gewöhnliches Ethernet-Interface, z.B. eth0. Aus Sicht des Host-Systems handelt es sich um ein so genanntes TAP-Interface, z.B. tap0. Beide Seiten des Interfaces haben eigene IP-Adressen, so dass für diese Art der Anbindung immer zwei IP-Adressen pro Maschine benötigt werden (ähnlich wie bei PPP): Eine IP-Adresse repräsentiert die virtuelle Maschine, die andere IP-Adresse ist dem

Host-System zugeordnet. Das virtuelle Netzwerk-Interface erlaubt nun primär nur den direkten Austausch von Netzwerk-Paketen zwischen den beiden Endpunkten (daher der Name). Um die virtuelle Maschine im lokalen Netz des Host-Systems sichtbar zu machen, muss eine Netzwerk-Route gesetzt und ein statischer ARP-Eintrag gemacht werden. Die eingebaute Setup-Funktion von UMLMON übernimmt diese Details.

Bei einem **virtuellen Netzwerk-Segment** wird die Bridging-Funktion des Linux-Kernels verwendet (diese muss in den Host-Kernel einkompiliert sein). Das virtuelle Segment verhält sich wie ein echtes Ethernet-Segment, an dem statt echter Netzwerk-Karten eben virtuelle Schnittstellen hängen. Man kann frei entscheiden, welche virtuellen Maschinen an welches Segment angeschlossen werden. Die virtuellen Segmente können auch mit echten Netzwerk-Karten des Host-Systems verbunden werden. Anders als bei Host-to-Host sind die virtuellen Segmente nicht auf IPv4 beschränkt; alle Ethernet-Protokolle können benutzt werden.

Die virtuellen Segmente sind sehr flexibel und leistungsfähig. Die Konfiguration dieser Option ist aber deutlich aufwändiger, da massiv in die Netzwerk-Setzungen des Host-Systems eingegriffen werden muss (Konfiguration von Bridging, Routen, Spanning Tree Protocol und ggf. Paketfilter). Dies ist nur etwas für erfahrene Linux- und Netzwerk-Anwender!

3 Installation von UMLMON

3.1 Installation der UMLMON-Software

UMLMON gibt es in drei Formaten:

- Als LSB-Paket. LSB steht für „Linux Standard Base“ und ist eine Technik, um Software zu distribuieren, die (potenziell) auf allen Linux-Distributionen eingesetzt werden kann. Leider unterstützen noch nicht alle Distributionen LSB. Falls Ihre Distribution LSB in Versionen 2 oder 3 unterstützt, sollten Sie UMLMON als LSB-Paket einspielen.

  Leider ist LSB in Version 3 nicht mit Version 2 kompatibel; aus diesem Grund gibt es für jede LSB-Version ein eigenes Paket.

  Das LSB-Paket hat den Dateinamen
  `lsb-gerd-stolpmann.de-umlmon-1.0.4-1.i386.rpm`
- Als übersetztes tar-Archiv mit Namen
  `umlmon-1.0.4-1.i386.tar.gz`
- Als Quellcode. Bitte haben Sie Verständnis, dass die Erläuterung, wie man den Quellcode übersetzt, dieses Handbuch sprengen würde.

Die Pakete und Archive erhalten Sie unter http://www.gerd-stolpmann.de/umlmon.

Nach der Installation (s.u.), werden Sie folgende Dateien und Verzeichnissen vorfinden:

- /usr oder /usr/local: Dieser Dateibaum wird die eigentlichen Programme enthalten.
- /etc/umlmon: Dies ist die zentrale Konfigurationsdatei.
- /etc/services: UMLMON wird einen Eintrag für den UMLMON-Verzeichnisdienst umldir und den Redirektionsport umlredir hinzufügen.
- /etc/init.d/umlmon: Dieses Skript dient zum Hochfahren während des Bootens
- /var/lib/umlmon: Dies ist der Datenbereich von UMLMON, der für die Aufnahme der virtuellen Maschinen vorgesehen ist.

Installation des LSB-Pakets auf einem RPM-basierten System

RPM ist der „Red Hat Package Manager“ und wird für die meisten Distributionen verwendet, um installierte Software zu verwalten.

Um die Installation nun vorzunehmen, gehen Sie bitte wie folgt vor:

1. Loggen Sie sich als Benutzer *root* in das Host-System ein und kopieren die rpm-Datei auf das Host-System (z.B. mit scp).
2. Führen Sie aus:
   `rpm -i lsb-gerd-stolpmann.de-umlmon-1.0.4-1.i386.rpm`

Ggf. funktioniert der rpm-Aufruf nicht und Sie müssen zuerst die Unterstützung für LSB installieren. Dies ist leider je nach Linux-Distribution unterschiedlich.

Falls Sie die Bridging-Funktion (virtuelle Netzwerk-Segmente) benutzen möchten, müssen Sie außerdem das bridge-utils-Paket installieren, das Ihrer Distribution beiliegt.

Installation des LSB-Pakets auf einem DEB-basierten System

DEB ist der Debian-Paketmanager und wird für Debian und die abgeleiteten Distributionen verwendet, um installierte Software zu verwalten.

Um die Installation nun vorzunehmen, gehen Sie bitte wie folgt vor:

1. Loggen Sie sich als Benutzer *root* in das Host-System ein und kopieren die rpm-Datei auf das Host-System (z.B. mit scp).
2. Führen Sie aus:
   `apt-get install lsb`
3. Führen Sie aus:
   `alien -i lsb-gerd-stolpmann.de-umlmon-1.0.4-1.i386.rpm`
4. Falls Sie die Bridging-Funktion (virtuelle Netzwerk-Segmente) benutzen möchten, führen Sie aus:
   `apt-get install bridge-utils`

Installation des TAR-Archivs

Um die Installation nun vorzunehmen, gehen Sie bitte wie folgt vor:

1. Loggen Sie sich als Benutzer *root* in das Host-System ein und kopieren die tar-Datei auf das Host-System (z.B. mit scp).
2. Führen Sie aus:
   ```
   cd /
   tar xzf umlmon-1.0.4-1.i386.tar.gz
   ```
3. Führen Sie das Post-Installationskript auf:
   ```
   /usr/local/sbin/umlmon-postinstall
   ```

Das Skript trägt UMLMON nicht als Dienst ein, der beim Booten des Systems automatisch gestartet wird. Die Methode hierfür unterscheidet sich je nach Linux-Distribution.

Debian: Rufen Sie das Kommando
```
update-rc.d umlmon defaults 99
```
auf.

SUSE: Rufen Sie das Kommando
```
insserv /etc/init.d/umlmon
```
auf.

3.2 Optional: Verlegen des Shared-Bereichs

UMLMON erlaubt die Existenz eines Verzeichnisses, das von allen virtuellen Maschinen aus zugänglich ist. Dieses Verzeichnis befindet sich nach Installation an dem Ort `/var/lib/umlmon/shared`. In diesem Verzeichnis können Dateien abgelegt werden, die von allen Maschinen benötigt werden, z.B. Gast-Kernel oder Referenz-Platten.

Falls der Standard-Ort ungeeignet ist, kann man ihn an eine andere Stelle des Dateisystems verlegen. Dazu editieren Sie bitte die Datei `/etc/umlmon`. Im Abschnitt `[GLOBAL]` finden Sie eine Zeile

```
shared = /var/lib/umlmon/shared
```

Ändern Sie den Pfad beliebig ab.

Wichtig: Das hier angegebene Verzeichnis muss existieren. Es darf nur für root beschreibbar sein.

Die Verlegung ist auch nachträglich möglich.

Im Rest der Anleitung nehmen wir weiter den Standard-Pfad an. Falls Sie den Shared-Bereich verlegen, müssen Sie die Pfade, die in der Anleitung genannt werden, ggf. eigenhändig abändern.

3.3 Installation von passenden Linux-Kerneln

Falls Sie den SKAS3-Modus verwenden wollen, installieren Sie jetzt bitte einen Linux-Kernel auf Ihrem Host-System, der den SKAS3-Patch enthält. (Sie können

diesen Schritt überspringen, wenn Sie UML nicht im SKAS3-Modus betreiben wollen.) In Anhang A finden Sie Hinweise, wie Sie dies bewerkstelligen können.

Den Gast-Kernel kopieren Sie bitte in das Verzeichnis

```
/var/lib/umlmon/shared/kernels
```

beispielsweise

```
cp .../linux /var/lib/umlmon/shared/kernels
```

Hinweis: Der Gast-Kernel ist in der Regel eine einzelne ausführbare Datei namens „linux“. Es ist, wie auch bei „normalen“ Kerneln, möglich, optionale Funktionen in Form von Kernel-Modulen zu übersetzen. Für den Einstieg raten wir davon jedoch ab, da dies die Installation deutlich komplizierter macht und die Vorteile gering sind.

3.4 Installation des Root-Images

Kopieren Sie nun das Dateisystem-Image, das Sie für die virtuellen Maschinen anfänglich verwenden möchten, in das Verzeichnis

```
/var/lib/umlmon/shared/disks
```

und geben Sie ihm einen Namen, der auf „.dsk“ endet, z.B.

```
cp .../root.dsk /var/lib/umlmon/shared/disks
```

In Anhang B finden Sie Hinweise, wie Sie ein solches Root-Image erzeugen können. Das Root-Image ist in der Regel ein ext2- oder ext3-Dateisystem, das als Wurzel-Dateisystem dem Gast-Kernel übergeben wird und in dem sich alle Dateien befinden, die zum Hochfahren der virtuellen Maschine benötigt werden.

Dieses Root-Image sollten Sie zu Beginn so klein wie möglich halten. Es dient zunächst nur zum Hochfahren eines Minimal-Systems, in dem Sie dann weitere Software nach Bedarf installieren können. Wir empfehlen, wir hier auch dargestellt, dieses Image als Vorlage nach `shared/disks` zu stellen, wo es von allen virtuellen Maschinen benutzt werden kann.

Hinweis: Viele Linux-Distributionen benutzen außerdem ein so genanntes `initrd`-Image (*initial ramdisk*). Diese Vorgehensweise ist nur sinnvoll, wenn ein modularer Gast-Kernel eingesetzt wird, und macht die Installation noch einmal deutlich komplizierter. Alle Linux-Distributionen können auch ohne `initrd`-Image hochgefahren werden, so dass wir davon abraten, diese Möglichkeit in Betracht zu ziehen.

3.5 Optional: Einrichtung eines gemeinsamen Benutzerkontos

Falls Sie sich dafür entschieden haben, für alle virtuellen Maschinen dasselbe

Konto zu verwenden, ist nun der richtige Zeitpunkt gekommen, dieses Konto auch einzurichten (im Host-System).

Je nach Linux-Distribution, die Sie verwenden, stehen unterschiedliche Werkzeuge zur Verfügung (z.B. „Computer → Systemkonfiguration → Benutzer und Gruppen“ bei Ubuntu, oder „yast2“ bei SuSE). Das Konto muss folgende Eigenschaften haben:

- Name: beliebig, z.B. „uml“
- Hauptgruppe: es sollte eine eigene Hauptgruppe angelegt werden, die genauso heißt wie das Konto, und in dem das Konto das einzige Mitglied ist
- Login-Eigenschaften: Es ist Ihnen überlassen, ob Sie ein Login zulassen wollen oder nicht. In ersterem Fall geben Sie ein Heimatverzeichnis, eine Shell und ein Passwort an. In letzterem Fall setzen Sie die Shell auf /bin/nologin (wenn von Ihrer Distribution angeboten) oder sonst auf /bin/false.
- Die numerische Kontonummer ist beliebig.

Beispiel Debian: Legen Sie das Konto mit dem Kommandoaufruf

```
adduser --system --home / --group uml
```

an.

Beispiel SUSE: Legen Sie das Konto mit den Kommandoaufrufen

```
groupadd --system uml
useradd --system --home / -g uml uml
```

an.

3.6 Starten von UMLMON

Solange Sie noch keine virtuelle Maschine konfiguriert haben, ist das Starten eigentlich entbehrlich. Ohne Maschine wird lediglich der Verzeichnisdienst gestartet. Sie können dies aber probehalber schon einmal tun mit:

```
/etc/init.d/umlmon start
```

Das Herunterfahren von UMLMON wird analog bewerkstelligt durch:

```
/etc/init.d/umlmon stop
```

Sobald Sie Maschinen konfiguriert haben, können Sie diese zur Menge der aktiven Maschinen hinzunehmen. Hierzu führen Sie aus:

```
/etc/init.d/umlmon update
```

Mit diesem Kommando können Sie auch Maschinen, die nicht mehr aktiv sein sollen, aus der Menge der aktiven Maschinen herausnehmen.

4 Einrichtung von virtuellen Maschinen

In diesem Kapitel wird gezeigt, wie man virtuelle Maschinen erzeugt und konfiguriert. Der gleich folgende Abschnitt über globale Konfigurationsoptionen kann übersprungen werden, da bei der Installation von UMLMON eine sinnvolle Ausgangskonfiguration angelegt wird.

4.1 Globale Konfigurationsoptionen

Folgende Optionen können in der Datei `/etc/umlmon` global für alle virtuellen Maschinen festgelegt werden. Diese Optionen müssen im Abschnitt `[GLOBAL]` stehen:

- **Erlauben von TCP-Verbindungen:** Wenn Sie per TCP-Verbindungen auf die UMLMON-Monitore zugreifen möchten, können Sie dieses Merkmal aktivieren, indem Sie die Option

  ```
  tcp = true
  ```

  setzen. Beachten Sie bitte, dass TCP-Verbindungen stets mit einem Passwort gesichert sein müssen (sonst ist keine Verbindung möglich). Beachten Sie bitte auch, dass Sie hiermit ein erhöhtes Sicherheitsrisiko eingehen, da die TCP-Verbindungen unverschlüsselt sind.

- **Erlauben von TCP-Redirektions-Verbindungen:** Eine zweite Möglichkeit, TCP-Verbindungen zu aktivieren, besteht in der Benutzung des Redirektionsports. Damit können alle Monitore eines Hosts über einen einzelnen Port angesteuert werden, was Vorteile für die Konfiguration von Firewalls bringt. Setzen Sie hierzu:

  ```
  tcp_redir = true
  ```

  Auch bei dieser Option müssen Passwörter verwendet werden.

- **Master-Passwort setzen:** Sie können ein Master-Passwort setzen, das anstelle der individuellen Passwörter zum Login über TCP verwendet werden kann. Dies kann in einigen Konfigurationen vorteilhaft sein. Erzeugen Sie zunächst den Hash-Wert des Passwortes, indem Sie

  ```
  /usr/sbin/umladmin password
  ```

  aufrufen. Sie werden nach dem Passwort gefragt, und es wird der Hash-Wert ausgegeben (ein Hex-String). Diesen Hash-Wert tragen Sie bitte in die Option

  ```
  password = <hashwert>
  ```

  ein.

Hinweis: Der Hash-Wert verschlüsselt das Passwort nicht! Bereits die Kenntnis des Hash-Wertes des Passworts reicht aus, um sich per TCP einloggen zu können. Der Hash-Wert schützt aber vor dem zufälligen Aufschnappen des Passworts durch unautorisierte Personen, und erhöht das Sicherheitsniveau beim Einsatz einer Administrationsshell wie umlwatch und UMLMON-Web.

- **Shared-Verzeichnis setzen:** Über die Option `shared` können Sie bestimmen, wo das Shared-Verzeichnis liegen soll (siehe Abschnitt 3.2). Wenn Sie die Option `shared` löschen, richtet UMLMON kein Shared-Verzeichnis ein.
- **Site-Kommando einrichten:** Über die Option `sitecmd` können Sie ein Skript angeben, das kundenspezifische Funktionserweiterungen enthält. Details finden Sie im Abschnitt 6.1.

Wenn Sie globale Optionen ändern, werden diese erst beim Hochfahren von Monitoren beachtet. Um Optionen sofort wirksam werden zu lassen, rufen Sie auf:

```
/etc/init.d/umlmon restart
```

Achtung: Dieses Kommando bewirkt, dass alle aktiven virtuellen Maschinen herunter- und wieder hochgefahren werden!

4.2 Vorbereitung: Benutzerkonten und Datenbereiche

Falls Sie für jede virtuelle Maschine ein eigenes Benutzerkonto verwenden möchten, legen Sie jetzt bitte die Konten für die Maschinen an.

Je nach Linux-Distribution, die Sie verwenden, stehen hierfür unterschiedliche Werkzeuge zur Verfügung (z.B. „Computer → Systemkonfiguration → Benutzer und Gruppen“ bei Ubuntu, oder „yast“ bei SuSE). Die Konten müssen folgende Eigenschaften haben:

- Name: beliebig, z.B. „vm1“ („vm“ für virtuelle Maschine). Es empfiehlt sich, für Konto und Maschine die gleichen Bezeichnungen vorzusehen.
- Hauptgruppe: es sollte eine eigene Hauptgruppe angelegt werden, die genauso heißt wie das Konto, und in dem das Konto das einzige Mitglied ist
- Login-Eigenschaften: Es ist Ihnen überlassen, ob Sie ein Login zulassen wollen oder nicht. In ersterem Fall geben Sie ein Heimatverzeichnis, eine Shell und ein Passwort an. In letzterem Fall setzen Sie die Shell auf /bin/nologin (wenn von Ihrer Distribution angeboten) oder sonst auf /bin/false.
- Die numerische Kontonummer ist beliebig.

Beispiel Debian: Legen Sie das Konto mit dem Kommandoaufruf

```
adduser --system --home / --group vm1
```

an.

Beispiel SUSE: Legen Sie das Konto mit den Kommandoaufrufen

```
groupadd --system vm1
useradd --system --home / -g vm1 vm1
```

an.

Falls Sie die Datenbereiche für die virtuellen Maschinen nicht im Dateibaum von `/var/lib/umlmon` haben möchten, präparieren Sie jetzt bitte ein alternatives Dateisystem, in dem ausreichend Platz ist.

4.3 Aufsetzen einer virtuellen Maschine

Virtuelle Maschinen werden durch zwei Schritte erzeugt:

1. Hinzunahme eines Konfigurationsabschnitts in `/etc/umlmon`
2. Starten des Monitors

Durch das Starten des Monitors wird nicht automatisch die virtuelle Maschine hochgefahren. Der Monitor kann nun aber über die Administrationsshells umlwatch und UMLMON-Web kontaktiert werden. Hierüber sind dann weitere Konfigurationsanpassungen und natürlich auch das Hoch- und Herunterfahren der virtuellen Maschine möglich.

In diesem Abschnitt wird eine einfache Standardkonfiguration für neue virtuelle Maschinen vorgeschlagen. Diese kann durch die Erläuterungen im weiteren Verlauf dieser Anleitung verfeinert werden.

Ein Konfigurationsabschnitt in `/etc/umlmon` hat die Form:

```
[Name]
Option1 = Wert1
Option2 = Wert2
...
```

Als Namen tragen Sie den Namen der neuen Maschine ein. Als Optionen empfehlen wir zu Beginn:

```
[Name]
vmuser = Benutzerkonto
logfile = monitor
start = manual
kernel = /shared/kernels/linux
mem = 64
ubd0 = /disks/root.cow
con0 = pty:con0
con1 = pty
con2 = pty
con3 = pty
con4 = pty
```

Mit `vmuser` legen Sie das Benutzerkonto fest. In `logfile` wird der Name der

Logdatei des Monitors angegeben. Die `start`-Option bestimmt, wann die Maschine hochgefahren werden soll (`manual`: nur auf Benutzerkommando). Die `kernel`-Option legt fest, welcher Gast-Kernel gestartet werden soll. In `mem` wird die Größe des virtuellen Hauptspeichers in MB angegeben. `ubd0` ist die virtuelle Festplatte, auf der sich das Betriebssystem befindet (d.h. das Root-Image; beachten Sie die Anweisungen weiter unten). Die `con`-Optionen bestimmen, wie mit den virtuellen Konsolen umgegangen werden soll. Auf `con0` werden Boot- und Log-Meldungen ausgegeben; `con1` und höher können für interaktive Logins benutzt werden. In dieser Konfiguration wird angegeben, dass `con0` bis `con4` durch den Monitor verwaltet werden sollen, und für `con0` ist eine Logdatei gleichen Namens angegeben.

Wie Sie sehen, haben wir als Root-Image eine Datei angegeben, die noch gar nicht existiert (`/disks/root.cow`). Das ist zunächst nicht schlimm, aber wir werden dafür sorgen müssen, dass diese Datei bis zum ersten Hochfahren der Maschine angelegt werden wird.

Beachten Sie, dass Log-Dateien normalerweise ohne Verzeichnispfad konfiguriert werden. Die Dateien werden dann in dem vorkonfigurierten Log-Verzeichnis angelegt.

Die Optionen `kernel` und `ubd0` enthalten Dateinamen, die merkwürdig anmuten: Die Verzeichnisse `/shared/kernels` und `/disks` existieren doch gar nicht! Hier kommt nun ins Spiel, dass die virtuelle Maschine in einem chroot-Jail gestartet werden wird. Dies bedeutet, dass eine spezielle Prozess-Umgebung aufgebaut wird, die auf das Jail-Verzeichnis `/var/lib/umlmon/Name/jail` beschränkt ist. Die Maschine kann nur Dateien aus diesem Verzeichnisbaum ansprechen, und zwar ist aus der Sicht der Maschine der Pfad `/shared/kernels` in Wirklichkeit /var/lib/umlmon/*Name*/jail/shared/kernels und der Pfad `/disks` in Wirklichkeit /var/lib/umlmon/*Name*/jail/disks. Im Referenz-Kapitel (Abschnitt 7.1) wird für jede Option angegeben, ob sich die Dateipfade „normal“ verhalten oder relativ zum Jail-Verzeichnis angegeben werden müssen.

Nachdem Sie den Konfigurationsabschnitt zu `/etc/umlmon` hinzugefügt haben, können Sie den Monitor starten:

```
/etc/init.d/umlmon update
```

(wenn Sie vorher, wie empfohlen, bereits `/etc/init.d/umlmon start` ausgeführt hatten - sonst tun Sie dies einfach jetzt).

Der Effekt ist nun:

- Der Monitor-Prozess läuft, und nimmt über den Socket /var/lib/umlmon/*Name*/ctrl Kommandos an (falls TCP eingeschaltet ist, außerdem über einen TCP-Port)

- Es wird eine Monitor-Logdatei geschrieben: `/var/lib/umlmon/`*`Name`*`/log/monitor`. Die Datei enthält eine Startup-Meldung. In dieser Datei finden Sie außerdem wichtige Hinweis- und Fehler-Meldungen des Monitors.
- Der Monitor hat einen so genannten Bind-Mount eingerichtet, der den `/shared`-Bereich auch innerhalb des Jails sichtbar macht, d.h. `/var/lib/umlmon/`*`Name`*`/jail/shared` ist ab jetzt ein existenter Pfad.

Um nun Kommandos zum Monitor zu schicken, rufen Sie umlwatch auf:

```
umlwatch Name
```

Sie können nun Kommandos eingeben, die vom Monitor ausgeführt werden, bis Sie die Shell beenden (mit dem Kommando `exit`).

Um nun die virtuelle Maschine endlich zu aktivieren, müssen Sie noch zwei Sachen erledigen:

- Erzeugung von `/disks/root.cow`: Das eigentliche Root-Image wurde ja als `/shared/disks/root.dsk` installiert. UMLMON betrachtet Images in `/shared` stets als schreibgeschützt, da diese von allen Maschinen parallel benutzt werden können. Für ihre ersten Schritte bietet es sich an, ein Copy-On-Write-Image zu benutzen, das `root.dsk` als Referenz verwendet. Dieses Image erzeugen Sie nun mit dem umlwatch-Kommando

  ```
  disk-create-cow /disks/root.cow:/shared/disks/root.dsk
  ```

- Hochfahren der virtuellen Maschine: Dies geschieht mit dem umlwatch-Kommando

  ```
  start
  ```

  Es ist wahrscheinlich, dass bei Ihren ersten Versuchen noch nicht alles funktioniert. Schauen Sie daher bitte in die Logdatei `/var/lib/umlmon/`*`Name`*`/log/monitor`, um zu sehen, ob der Gast-Kernel gebootet werden konnte.

Falls die Maschine nicht hochfährt, können Sie die maschinen-spezifischen Optionen in `/etc/umlmon` direkt verändern und sofort noch einmal das `start`-Kommando probieren, da bei jedem Startversuch die Datei `/etc/umlmon` neu eingelesen wird.

Wichtige weitere Kommandos von umlwatch:

- Kommando `info`: Dieses gibt die aktuelle Konfiguration und den aktuellen Zustand der Maschine aus. Sie können hier auch unmittelbar sehen, ob die Maschine läuft oder inaktiv ist.
- Kommando `tail monitor`: Gibt die letzten 20 Zeilen des Monitor-Logs aus.
- Kommando `tail con0`: Gibt die letzten 20 Zeilen des Konsolen-Logs aus. Hier

sollten Sie die üblichen Boot-Meldungen des Linux-Kernels finden.

- Kommando `ctrlaltdel`: Fährt die Maschine herunter. Hierzu wird ein so genannter CTRL-ALT-DEL-Request erzeugt und an den laufenden Gast-Kernel geschickt. Das Gast-Betriebssystem muss so konfiguriert sein, dass dieser Request zum Herunterfahren führt (und nicht zum Neustart). Wie Sie dies einrichten können, wird in Abschnitt B.1 beschrieben.
- Wenn CTRL-ALT-DEL noch nicht funktioniert, können Sie das Gast-System auch relativ hart stoppen, indem Sie folgende drei Kommandos nacheinander eingeben:
  ```
  sysrq s
  sysrq u
  halt
  ```
  Durch diese Methode werden die Shutdown-Skripte jedoch nicht ausgeführt; für den produktiven Betrieb ist sie daher ungeeignet.
- Kommando `connect con1`: Wenn das Gast-System hochgefahren ist, können Sie sich mit diesem Kommando interaktiv einloggen. Das Gast-System gibt einen Login-Prompt aus, und Sie können sich mit Kontoname und Passwort anmelden. Wenn Sie ihre Sitzung wieder beenden wollen, loggen Sie sich zunächst aus, und drücken dann die Tasten Strg-AltGr-9 (auf einer deutschen Tastatur, entsprechend dem Steuerzeichen CTRL-]). Damit lösen Sie die Verbindung mit der Gast-Konsole und kommen wieder zur umlwatch-Shell zurück.

  Hinweise: Wenn Sie sich nicht ausloggen, bleiben Sie eingeloggt und setzen die Sitzung mit dem nächsten `connect`-Kommando fort.

  Falls Sie sich von einem xterm-Emulator (oder gleichwertig, z.B. Gnome-Terminal oder KDE-Konsole) aus einloggen, sollten Sie als erstes Kommando eingeben:
  ```
  export TERM=xterm
  ```
  Ansonsten stimmen die Terminal-Steuerzeichen nicht, und bei einigen Programmen sehen Sie nur „Zeichensalat“ auf dem Bildschirm.

4.4 Optional: Verlegen des Datenbereichs

Normalerweise befinden sich die Daten der virtuellen Maschinen in dem Verzeichnis `/var/lib/umlmon`. Sie können ein Teil dieser Daten in ein anderes Dateisystem verlegen, z.B. weil Sie dort ausreichend Platz haben.

Die Verzeichnisstruktur ist wie folgt: Für eine Maschine gibt es folgende Dateien und Verzeichnisse:

- `/var/lib/umlmon/`*Name*`/ctrl`: Dies ist der Steuer-Socket (nur vorhanden bei aktivem UMLMON). Er kann nicht verlegt werden.

- `/var/lib/umlmon/`*`Name`*`/config`: Dies ist die vom Benutzer geänderte Konfiguration der virtuellen Maschine. Diese Datei kann ebenfalls nicht verlegt werden.
- `/var/lib/umlmon/`*`Name`*`/jail`: Dies ist das Jail-Verzeichnis, in dem sich die virtuellen Platten und weitere Laufzeit-Dateien befinden. Dieses Verzeichnis kann pro Maschine durch eine `jaildir`-Direktive an einen anderen Ort verlegt werden.
- `/var/lib/umlmon/`*`Name`*`/log`: Dies ist das Log-Verzeichnis, in dem sich alle von UMLMON für die jeweilige Maschine geschriebenen Log-Dateien befinden. Dieses Verzeichnis kann pro Maschine durch eine `logdir`-Direktive an einen anderen Ort verlegt werden.

Die jaildir- und logdir-Optionen tragen Sie bitte in den Konfigurationsabschnitt für die jeweilige Maschine ein, für die sie gelten sollen, z.B.

```
[Name]
jaildir = Verzeichnis
logdir = Verzeichnis
...
```

Diese Optionen werden dann beim nächsten Hochfahren der Maschine aktiv. Zu diesem Zeitpunkt sollten Sie die Plattendateien an ihren neuen Ort kopiert haben.

Hinweis: Kopieren Sie Plattendateien bitte nur mit `cp -p` oder `cp -a`, d.h. unter Beibehaltung der Dateiattribute. Ansonsten kann es passieren, dass User Mode Linux Dateien zurückweist. **Dieses Problem ist später nur schwer zu korrigieren!**

4.5 Optional: Netzwerkkonfiguration

In diesem Abschnitt stellen wir die Host-to-Host-Konfiguration von Netzwerken vor. Virtuelle Netzwerk-Segmente werden in Abschnitt 6.2 behandelt. Generell sollten Sie Netzwerke erst dann konfigurieren, wenn die virtuelle Maschine bereits grundsätzlich läuft und Sie sich einloggen können.

Voraussetzung für Netzwerke ist, dass im Host-Kernel die TUN/TAP-Treiber verfügbar sind. Dies sollte normalerweise der Fall sein. Falls diese Treiber als Modul übersetzt sind, versucht UMLMON, das entsprechende Modul automatisch zu laden.

Für ein Host-to-Host-Netzwerk benötigen Sie zwei IP-Adressen. Wir nehmen im Folgenden an, dass der Host sich in einem lokalen Netz befindet, und dass `eth2` die Schnittstelle zu diesem Netz ist. Die zwei neuen IP-Adressen sollten aus dem gleichen Adressbereich kommen, aus dem auch `eth2` seine Adresse bezieht.

> **Beispiel:** Wenn `eth2` zu einem Klasse-C-Netzwerk 192.168.13.0/24 gehört, sollten die beiden neuen Adressen auch aus diesem Adressbereich stammen.

Wir nehmen an, Ihr Netzwerk-Administrator weist Ihnen die Adressen 192.168.13.70 und 192.168.13.71 zu.

Wenn Sie Adressen aus einem anderen Bereich nehmen, werden Sie Schwierigkeiten bekommen, die virtuelle Maschine aus dem LAN zu erreichen.

Die Host-to-Host-Konfiguration geschieht über ein virtuelles Netzwerk-Interface, das sowohl vom Host- als auch vom Gast-System aus angesprochen werden kann. Im Host-System ist dieses Interface unter dem Namen `tap`*N* bekannt, wobei *N* eine Nummer ist. Im Gast-System wird dieses Interface den Namen `eth0` haben. Da `tap`*N* ein host-seitiger Stellvertreter für das Interface des Gast-Systems ist, nennen wir es auch Proxy-Interface.

Beispiel: Sie haben bereits vier Maschinen konfiguriert, die die Interfaces `tap0` bis `tap3` haben. Für ihre neue Maschine verwenden Sie daher `tap4`.

Nun haben Sie alle Informationen zusammen, um das Netzwerk konfigurieren zu können. Fügen Sie folgende Zeile (ohne Umbrüche) zu dem Konfigurationsabschnitt der Maschine in /etc/umlmon hinzu:

```
eth0 = host_to_host(proxy_if=ProxyIF,
                    proxy_addr=HostIP,
                    guest_addr=GastIP,
                    host_if=HostIF)
```

In unserem Beispiel:

```
eth0 = host_to_host(proxy_if=tap4,
                    proxy_addr=192.168.13.71,
                    guest_addr=192.168.13.70,
                    host_if=eth2)
```

In der Regel ist dies bereits UMLMON-seitig alles. Sie können die virtuelle Maschine neu starten. Auf dem Host-System wird nun das neue Interface `tap4` erscheinen, und im Gast-System ist `eth0` verfügbar. `tap4` müssen Sie nicht weiter konfigurieren, dies hat UMLMON bereits für Sie erledigt.

Hinweise: Es wird in die Routing-Tabelle des Host-Systems eine Host-Route eingetragen, und es wird ein Eintrag in die ARP-Tabelle gemacht. Dadurch ist die virtuelle Maschine in dem LAN-Segment, zu dem auch das Host-System gehört, sichtbar wie ein echter Rechner.

Falls Sie auf dem Host-System einen Paketfilter installiert haben (iptables), müssen Sie ggf. neue Regeln für das tap-Interface hinzunehmen.

Im Gast-System ist eth0 allerdings noch nicht konfiguriert. Sie können `eth0` wie eine normale Ethernet-Karte verwenden, und daher dieses Interface mit den Mitteln konfigurieren, die Ihre Linux-Distribution hierfür zur Verfügung stellt.

Beispiel Debian: Fügen Sie zu `/etc/network/interfaces` folgende Zeilen hinzu:

```
auto eth0
iface eth0 inet static
 address 192.168.13.71
 netmask 255.255.255.255
 up route add -host 192.168.13.70 dev eth0
 up route add -net 0.0.0.0 gw 192.168.13.70 dev eth0
```

Aktivieren Sie das Interface mit dem Kommando:

```
ifup eth0
```

Beispiel SUSE: Am einfachsten konfigurieren Sie das Interface mit Hilfe von yast2 (Netzwerkgeräte → Netzwerkkarte).

5 Einrichtung von Benutzerzugängen

5.1 Wie funktioniert der lokale Zugang?

Der Monitor kann lokal (d.h. vom selben Host aus) über einen Unix Domain Socket kontaktiert werden. Dieser Socket existiert als Datei:

`/var/lib/umlmon/`*`Name`*`/ctrl`

Dies ist wichtig zu wissen, da über die Dateirechte die Zugangsberechtigung ausgedrückt wird (siehe folgenden Abschnitt).

Über diesen Socket können alle administrativen Funktionen des Monitors, die für Benutzer möglich sind, aufgerufen werden. Dies sind in etwa die Kommandos, die über die umlwatch-Shell abgesetzt werden können.

5.2 Berechtigungen für den lokalen Zugang

Es kann geregelt werden, welche Linux-Konten den Socket kontaktieren können. Dies wird über das x-Bit des Verzeichnisses `/var/lib/umlmon/`*`Name`* ausgedrückt. Wenn für ein Konto das x-Bit gesetzt ist, ist der Kontakt möglich, sonst nicht.

Beispiel: Das Verzeichnis gehört dem Konto `vm5` und der Gruppe `customer3`. Die Berechtigungs-Bits sind auf -rwx--x--- gesetzt. Dies bedeutet: Das Konto `vm5`, dem die Maschine ohnehin gehört, und alle Konten der Gruppe `customer3` haben administrativen Zugang. Das Konto `vm5` kann außerdem die Dateien in diesem Verzeichnis direkt lesen und schreiben. Dies ist optional und für den reibungslosen Betrieb nicht erforderlich.

Der Socket `/var/lib/umlmon/`*`Name`*`/ctrl` wird jedes Mal, wenn der Monitor gestartet wird, neu erzeugt. Es ist daher sinnlos, die Berechtigung des Sockets selbst zu setzen.

Wenn Das Verzeichnis von UMLMON selbst erzeugt wird, werden die Berechtigungen so gesetzt, dass nur das Konto, dem die Maschine gehört, Zugang

hat (neben `root`).

5.3 Optional: Restriktion von ssh auf eine umlwatch-Shell

Es ist möglich, den ssh-Zugang eines Kontos so zu konfigurieren, dass nur das umlwatch-Kommando für eine bestimmte Maschine aufgerufen werden kann. Auf diese Weise kann den Benutzern direkter Zugang zu umlwatch gewährt werden, ohne gleichzeitig die beliebige Ausführung von Kommandos zu erlauben.

Genauer gesagt, wird nur der Zugriff mittels eines bestimmten öffentlichen Schlüssels restringiert. Hierzu ist die Datei `$HOME/.ssh/authorized_keys` zu editieren. Jede Zeile dieser Datei nennt einen öffentlichen Schlüssel, mit dessen Hilfe man sich ohne Passwort einloggen kann. Man kann auch Optionen angeben. Die Zeile muss folgendes Format haben, damit die Restriktion greift:

`command="/usr/bin/umlwatch` *`Name`*`" no-port-forwarding` *`Schlüssel`*

Dabei ist *`Schlüssel`* durch den öffentlichen ssh-Schlüssel zu ersetzen.

5.4 Wie funktioniert der TCP-Zugang?

Wenn der TCP-Zugang freigeschaltet ist (Option `tcp=true` ist im `GLOBAL`-Abschnitt gesetzt), richtet der Monitor zusätzlich zum lokalen Socket auch einen TCP-Socket ein. Über diesen Socket kann der Monitor von allen Rechnern kontaktiert werden, die das Host-System per TCP kontaktieren können.

Es wird eine Port-Nummer aus dem anonymen Bereich allokiert. Der Port wird stets an alle IPv4-Netzwerk-Interfaces gebunden.

Für eine Kontaktaufnahme wird stets ein Passwort vorausgesetzt. Dies kann entweder das Passwort sein, das für die Maschine konfiguriert ist, oder das Master-Passwort. Wenn kein Passwort vergeben ist, ist keine Kontaktaufnahme möglich.

Da ein anonymer Port verwendet wird, kann die Portnummer nicht vorausgesagt werden. Damit man sich dennoch über den Namen der virtuellen Maschine mit ihr verbinden kann, gibt es den UMLMON-Verzeichnisdienst umldir. Dieser kann über eine feste Portnummer kontaktiert werden (ohne Passwort), und über umldir lässt sich die Portnummer des Monitors, von dem nur der Name bekannt ist, in Erfahrung bringen.

> **Hinweis:** Die Portnummer von umldir wird über `/etc/services` festgelegt.

Die Administrationsshell umlwatch erlaubt den Zugriff auf einen Monitor, der auf einem fremden Rechner läuft, über die Aufruf-Form

> `umlwatch` *`Name@Host`*

Das Passwort wird interaktiv erfragt.

Wichtige Sicherheitshinweise: Ein TCP-Port öffnet konzeptbedingt ein

zusätzliches Einfallstor für Eindringlinge, die sich unautorisiert Zugriff auf das Host- oder ein Gast-System verschaffen wollen. Aus diesem Grunde ist es unverzichtbar, zusätzliche administrative Maßnahmen zu treffen, um die TCP-Ports zu schützen.

Das Sicherheitsniveau der TCP-Verbindungen wird wie folgt charakterisiert:

- Über die TCP-Verbindungen werden grundsätzlich alle RPC-Anfragen bis zu einer Länge von 64 KB entgegengenommen. Längere Anfragen werden als Denial-Of-Service-Attacke eingestuft und abgewehrt.
- Die Passwort-Überprüfung erfolgt per MAC (Message Authentication Code) und gilt als sicher. Die Passwort-Überprüfung erfolgt nur einmal pro TCP-Verbindung. Wenn es Angreifern möglich ist, bestehende TCP-Verbindungen zu kapern (TCP Spoofing), kann jedoch unautorisierter Zugang erlangt werden.
- Die Zahl gleichzeitiger TCP-Verbindungen ist auf 100 pro Monitor limitiert. Bei fehlgeschlagener Authentisierung wird eine Pause von 5 Sekunden erzwungen. Durch diese beiden Maßnahmen wird die maximale Frequenz von Wörterbuch-Attacken auf ca. 20 pro Sekunde begrenzt. Ein darüber hinausgehender Schutz gegen Wörterbuch-Attacken ist nicht vorgesehen.
- Die TCP-Verbindungen sind unverschlüsselt. Es wird explizit darauf hingewiesen, dass die übertragenen Nutzdaten wiederum Passwörter, Zugangscodes und andere sensitive Informationen enthalten können und dass kein Schutz gegen unbefugtes Mitlesen existiert.
- Für den Verzeichnisdienst existiert kein Passwortschutz. Über den Verzeichnisdienst kann somit jeder herausfinden, welche Maschinen existieren, und über welche Ports diese angesprochen werden können.

Best Practice: Es ist unvertretbar, dass die TCP-Ports über öffentliche Netze direkt zugänglich sind. Wenn ein Zugriff über öffentliche Netze notwendig ist, muss eine zusätzliche Sicherheitsschicht eingerichtet werden, die die TCP-Verbindungen verschlüsselt (z.B. über SSL). Dies kann z.B. über externe Zusatz-Software geschehen (z.B. stunnel).

Es ist vertretbar, wenn die TCP-Ports nur von Rechnern kontaktiert werden können, die als vertrauenswürdig eingestuft werden, z.B. weil sie der selben administrativen Kontrolle unterliegen. Die Begrenzung der Kontaktmöglichkeiten kann durch netzwerktechnische Maßnahmen (geschützte LANs) oder über Paketfilter geschehen.

> **Anmerkungen:** Die Kontaktmöglichkeit per TCP wurde vor allem deshalb in den Funktionsumfang von UMLMON aufgenommen, um eine Steuerung über die Web-Shell UMLMON-Web zu erlauben. Es ist dadurch ausreichend, dass UMLMON-Web auf einem einzelnen Rechner installiert ist, um so alle Maschinen des lokalen Netzes von einem Punkt aus administrieren zu können. Die Sicherheitsmaßnahmen wurden für diesen Einsatzzweck entworfen.

5.5 Variante: Der TCP-Redirektionsport

Anstelle anonymer Ports kann auch ein Redirektionsport vereinbart werden. Über diesen Port können alle Monitore eines Hosts kontaktiert werden. Diese Möglichkeit wird benutzt, wenn die Option `tcp_redir=true` im `GLOBAL`-Abschnitt gesetzt ist (dies kann zusätzlich oder alternativ zu den „normalen“ TCP-Ports geschehen).

Die zentrale Portnummer wird in `/etc/services` festgelegt (unter der Bezeichnung `umlredir`).

Die Redirektion wird über den Verzeichnisdienst umldir abgewickelt. Das Verfahren ist ebenfalls über Passwörter geschützt und genauso sicher wie der direkte TCP-Zugriff.

Für die Clients wie umlwatch ergibt sich keine Änderungen. Diese erkennen automatisch, ob der Redirektionsport aktiviert ist und benutzen ihn in diesem Fall.

5.6 Passwort-Management

Da die TCP-Ports Passwörter voraussetzen, gibt es folgende Management-Funktionen:

Setzen eines Master-Passworts: Dies ist bereits in Abschnitt 4.1 beschrieben.

Setzen des Passwortes einer virtuellen Maschine: Hierzu ist die Option `password` im für die Maschine zuständigen Abschnitt der Datei `/etc/umlmon` zu setzen. Falls `/var/lib/umlmon/`*`Name`*`/config` existiert, hat die Option `password` in dieser Datei Vorrang (siehe Abschnitt 7.1 für Erläuterungen zu dieser Datei). Die Option `password` muss jeweils auf den Hash-Wert des Passworts gesetzt werden. Der Hash-Wert ist eine Hex-Zahl, die über das Kommando

```
/usr/sbin/umladmin password
```

ermittelt werden kann.

Ändern des Passwortes einer virtuellen Maschine: Dies geschieht interaktiv über das Kommando `change-password` von umlwatch. Es ist nur möglich, das Passwort zu setzen, wenn das bisherige Passwort bekannt ist. Es ist mit dieser Funktion nicht möglich, ein Passwort erstmalig zu setzen. Die Änderung des Passworts ist auch über TCP-Verbindungen möglich; das neue Passwort wird dabei gesichert übertragen.

Löschen von Passwörtern: Um das Master-Passwort zu löschen, entfernen Sie die Option `password` aus dem `GLOBAL`-Abschnitt von `/etc/umlmon`. Um das Passwort einer virtuellen Maschine zu löschen, entfernen Sie die Option `password` sowohl aus dem entsprechenden Abschnitt von `/etc/umlmon` als auch aus der Datei `/var/lib/umlmon/`*`Name`*`/config` (wenn existent).

6 Fortgeschrittene Funktionen

6.1 Einrichten eines Site-Kommandos

Das Site-Kommando ist ein Skript, das der Host-Administrator hinterlegen kann, und das die Benutzer der virtuellen Maschinen aufrufen können. Dieses Skript ist vor allem für Aktionen gedacht, die praktisch immer spezielle Anpassungen an die lokale Umgebung benötigen. In umlwatch kann man das Skript mit dem Kommando `site` aufrufen. Die Oberfläche UMLMON-Web hat zwei Funktionen, die das Site-Kommando aufrufen: Duplizieren einer virtuellen Maschine und Löschen einer virtuellen Maschine. Die eigentliche Duplizierung bzw. Löschung wird nur durch das Skript durchgeführt und kann an die lokalen Verhältnisse angepasst werden (z.B. Programmierung einer automatischen Netzwerkkonfiguration oder automatische Sicherung gelöschter Maschinen). In der Quellcode-Distribution von UMLMON ist das Beispielskript `umlsite` enthalten, das eine einfache Duplizierung und Löschung implementiert.

> **Hinweise:** Das Skript läuft nur ordentlich, wenn für das Jail-Verzeichnis und das Log-Verzeichnis die Standardpfade verwendet werden. Das Skript editiert `/etc/umlmon`.

Das Site-Kommando muss stets durch einen Eintrag im `GLOBAL`-Abschnitt von `/etc/umlmon` aktiviert werden: Setzen Sie den Parameter `sitecmd` auf den absoluten Pfad des Skripts.

Aus umlwatch kann man die Funktionen dieses Skripts aufrufen durch:

- `site clone` *`NeuerName`*: Die aktuelle Maschine wird unter dem angegebenen Namen dupliziert
- `site delete`: Die aktuelle Maschine wird gelöscht.

Das Skript läuft als root-Benutzer ab! Damit kann das Skript beliebige Änderungen an der Konfiguration durchführen.

> **Hinweis:** Damit ist auch ein Sicherheitsrisiko verbunden. Das Skript muss absolut fehlerfrei programmiert sein, um Risiken zu vermeiden.

Das Skript hat folgende Aufrufkonventionen:

- Argumente werden als normale Kommandoargumente übergeben. Das erste Argument soll verwendet werden, um die Funktion aus ggf. mehreren Funktionen auszuwählen (z.B. „clone“ oder „delete“).
- Folgende Environment-Variablen sind gesetzt:
  - `UMLMON_ID`: Der Name der aktuellen virtuellen Maschine
  - `UMLMON_VARDIR`: Das Basisverzeichnis, i.d.R. `/var/lib/umlmon`

- UMLMON_LOGDIR: Das Log-Verzeichnis der aktuellen virtuellen Maschine
- UMLMON_JAILDIR: Das Jail-Verzeichnis der aktuellen virtuellen Maschine
- UMLMON_STATUS: Der Status der aktuellen virtuellen Maschine (INACTIVE, STOPPED, STARTING oder RUNNING)
- UMLMON_USER: Das Benutzerkonto, das der aktuellen virtuellen Maschine zugeordnet ist

- Ausgaben auf stdout oder stderr werden dem Benutzer angezeigt (allerdings mit Zeitverzögerung).
- Der Exit-Code 0 signalisiert Erfolg; andere Codes signalisieren Fehler.
- Das Skript kann im Prinzip beliebig lange laufen. Der Monitor ist währenddessen allerdings nicht blockiert; ein Benutzer kann eine zweite umlwatch-Sitzung eröffnen und parallel Kommandos absetzen.

6.2 Einrichtung eines virtuellen Netzwerk-Segments

Um diese Funktion zu nutzen, benötigen Sie einen Host-Kernel, der Ethernet-Bridging unterstützt. Außerdem benötigen Sie das Konfigurations-Tool `brctl`.

> **Debian und SUSE:** `brctl` ist Teil des Pakets `bridge-utils`.

Zunächst zum Konzept. Das Netzwerk-Segment besteht aus folgenden Komponenten:

- Für jede virtuelle Maschine, die angeschlossen werden soll, benötigen Sie ein Proxy-Interface.
- Wenn Sie das Host-System ebenfalls anschließen wollen (dies ist optional), so haben Sie zwei Möglichkeiten: (1) Sie können dem Host-System eine weitere IP-Adresse geben. In diesem Fall bleibt das virtuelle Segment getrennt von Ihrem LAN, es sei denn, Sie konfigurieren explizite Routen. (2) Sie können eine echte Netzwerk-Karte an das Segment anschließen und damit das virtuelle Segment mit dem echten LAN-Segment verbinden (oder überbrücken - daher der Name Bridging). Die echte Netzwerk-Karte muss allerdings den *promiscuous mode* unterstützen; nicht alle Karten tun dies (insbesondere nicht alle WLAN-Karten). Die Überbrückung funktioniert auch für Nicht-IP-Protokolle.
- Das virtuelle Segment erscheint selbst als Netzwerk-Interface mit Namen `br`*N*.

Wir erläutern zunächst den Fall, dass das Host-System mit einer zusätzlichen IP-Adresse an das virtuelle Segment angeschlossen wird.

Gehen Sie nun wie folgt vor:

1. Erzeugen Sie die Bridge:

   ```
   brctl addbr br0
   ```

2. Konfigurieren der Bridge:

```
brctl stp br0 off
brctl setfd br0 1
brctl sethello br0 1
```

   Hinweis: Mit dem ersten Kommando wird das Spanning Tree Protocol ausgeschaltet. Sie benötigen dieses nur, wenn Sie mehr als zwei LAN-Segmente haben, die über Bridges oder Switches verbunden sind. In diesem Beispiel ist das virtuelle Segment überhaupt nicht mit einem anderen Segment verbunden (eine geroutete Verbindung zählt nicht).

3. Das Bridge-Interface `br0` wird nun als zusätzliches Interface des Host-Systems konfiguriert:

```
ifconfig br0 IP netmask MASK up
```

   Ersetzen Sie *IP* durch eine IP-Adresse und *MASK* durch die Netzwerk-Maske.

4. In `/etc/umlmon` setzen Sie für jede Maschine (ohne Zeilenumbrüche):

```
eth0 = bridged(bridge_if=br0,
               proxy_if=ProxyName,
               proxy_hwaddr=pool,
               guest_hwaddr=pool)
```

   Dabei ersetzen Sie *ProxyName* durch einen sprechenden Namen für das Proxy-Interface.

5. Starten Sie die virtuellen Maschinen. Geben Sie `eth0` eine IP-Adresse, die aus dem gleichen Adressbereich stammt wie die Adresse von `br0`.

Die virtuellen Maschinen und das Host-System können nun miteinander kommunizieren. Falls Sie auf dem Host-System die Routing-Tabelle konfigurieren, können Sie die Maschinen sogar mit dem LAN verbinden. Das Host-System ist dann das Gateway für das virtuelle Segment.

Alternative: Sie müssen `br0` keine IP-Adresse geben, wenn Sie das Host-System nicht an das Segment anschließen wollen. Sie können auch andere Protokolle als IP benutzen.

Alternative: Wenn Sie das virtuelle Segment direkt mit dem LAN-Segment überbrücken wollen, gehen Sie wie folgt vor:

- Angenommen, eth2 ist die Netzwerk-Karte, um die es geht. Schalten Sie die Karte zunächst in den promisken Modus:

```
ifconfig eth2 0.0.0.0 promisc up
```

- Fügen Sie eth2 zur Bridge hinzu:

```
brctl addif br0 eth2
```

- Konfigurieren Sie br0 nun mit der IP-Adresse, die zuvor eth2 hatte (und nun

nicht mehr hat):

`ifconfig br0 IP netmask MASK up`

Debian: Mittels Einträgen in /etc/network/interfaces können Sie die gesamte Konfiguration der Bridge vornehmen. Damit brauchen Sie keine Skripte zu schreiben, die die Netzwerk-Konfiguration vornehmen. Details hierzu sprengen allerdings den Rahmen dieser Anleitung.

7 Referenz

7.1 Die Konfigurationsdatei /etc/umlmon

Diese Datei hat das bekannte INI-Format und ist in Abschnitte unterteilt:

```
[Abschnitt]
Parametername = Parameterwert
...

[Abschnitt]
Parametername = Parameterwert
...
```

Der Abschnitt GLOBAL enthält Parameter, die für alle virtuellen Maschinen gelten. Alle anderen Abschnitte konfigurieren die virtuelle Maschine mit dem Namen des Abschnitts.

Neben der Datei /etc/umlmon kann es pro virtueller Maschine noch eine weitere Datei mit Parametern geben, die die dynamische Konfiguration enthält. Diese Datei hat den Namen /var/lib/umlmon/*Name*/config. Wenn ein Benutzer mittels umlwatch oder UMLMON-Web Parameter ändert, werden diese nicht nach /etc/umlmon geschrieben, sondern in diese zusätzliche Datei. Wenn diese Datei existiert, haben alle Parameter, die in dieser Datei vorkommen können, Vorrang vor den Definitionen in /etc/umlmon.

Folgende Parameter können in GLOBAL benutzt werden (siehe auch Abschnitt 4.1):

- tcp: Bestimmt, ob der TCP-Port geöffnet wird (true) oder nicht (false). Erläuterungen finden Sie in Abschnitt 5.4.
- tcp_redir: Bestimmt, ob der TCP-Redirektionsport geöffnet wird (true) oder nicht (false). Erläuterungen finden Sie in Abschnitt 5.5.
- password: Das Master-Passwort als Hex-codierter MD5-Hash-Wert. Eine solche Kodierung können Sie mit dem umladmin-Kommando erzeugen (siehe Abschnitt 7.2). Das Master-Passwort kann für die Authentisierung einer TCP-Verbindung anstelle des Passworts der einzelnen virtuellen Maschine benutzt werden.
- shared: Der absolute Pfad zum Shared-Verzeichnis. Das Shared-Verzeichnis

wird, falls dieser Parameter gesetzt ist, in das Jail-Verzeichnis gemountet. Erläuterungen finden Sie in Abschnitt 3.2.

- `sitecmd`: Der absolute Pfad zum Site-Kommando. Erläuterungen finden Sie in Abschnitt 6.1.

Folgende Parameter können in einem Abschnitt für eine virtuelle Maschine benutzt werden. Viele dieser Parameter können auch in der dynamischen Konfigurationsdatei der virtuellen Maschine vorkommen. Wenn dies erlaubt ist, hat der Wert der dynamischen Konfiguration Vorrang.

- `start`: Gibt an, wann der Monitor gestartet und ggf. die virtuelle Maschine hochgefahren wird. Mögliche Werte:
    - `off`: Der Monitor wird durch das Boot-Skript `/etc/init.d/umlmon` sowie durch „`umladmin start`“ nicht gestartet.
    - `manual`: Der Monitor wird durch das Boot-Skript `/etc/init.d/umlmon` sowie durch „`umladmin start`“ gestartet, aber die virtuelle Maschine wird nicht automatisch hochgefahren. (Dies ist der Default-Wert.)
    - `boot`: Der Monitor wird durch das Boot-Skript `/etc/init.d/umlmon` sowie durch „`umladmin start`“ gestartet, und die virtuelle Maschine wird automatisch hochgefahren.

  Der Parameter `start` darf in der dynamischen Konfiguration gesetzt werden.
- `jaildir`: Dieser Parameter gibt an, wo das Jail-Verzeichnis liegt. Wenn Sie diesen Parameter setzen, achten Sie bitte darauf, dass das angegebene Verzeichnis existiert und eine geeignete Struktur hat. Wenn der Parameter nicht gesetzt wird, nimmt UMLMON `/var/lib/umlmon/`*`Name`*`/jail` als Jail-Verzeichnis und sorgt selbst dafür, dass die benötigten Unterverzeichnisse existieren. Dieser Parameter wird nur in `/etc/umlmon` akzeptiert und nicht in der dynamischen Konfiguration. Der Parameter muss ein absoluter Pfad sein.
- `logdir`: Dieser Parameter gibt an, wo das Log-Verzeichnis liegt. Wenn Sie diesen Parameter setzen, achten Sie bitte darauf, dass das angegebene Verzeichnis existiert. Wenn der Parameter nicht gesetzt wird, nimmt UMLMON `/var/lib/umlmon/`*`Name`*`/log` als Log-Verzeichnis. Dieser Parameter wird nur in `/etc/umlmon` akzeptiert und nicht in der dynamischen Konfiguration. Der Parameter muss ein absoluter Pfad sein.
- `logfile`: Der Name der Monitor-Logdatei, die im Log-Verzeichnis angelegt wird. Der Name soll keinen Verzeichnispfad enthalten. Wenn der Parameter nicht angegeben wird, so wird keine Logdatei geführt. Der Parameter darf in der dynamischen Konfiguration gesetzt werden.
- `vmuser`: Das Benutzerkonto, unter dem die virtuelle Maschine ausgeführt wird. Dieser Parameter muss angegeben werden. Dieser Parameter wird nur

in /etc/umlmon akzeptiert und nicht in der dynamischen Konfiguration.

- password: Das Passwort als Hex-codierter MD5-Hash-Wert. Eine solche Kodierung können Sie mit dem umladmin-Kommando erzeugen (siehe Abschnitt 7.2). Das Passwort dient zur Authentisierung des Benutzers, wenn der Monitor über eine TCP-Verbindung administriert wird. Das Passwort kann per umlwatch-Kommando change-password geändert, aber nicht gesetzt werden. Der Parameter darf in der dynamischen Konfiguration gesetzt werden.
- kernel: Der Pfad des Kernel-Executables. Dieser Pfad muss relativ zum Jail-Verzeichnis, aber mit führendem Schrägstrich angegeben werden (z.B. als /linux, wenn das Executable in Wirklichkeit /var/lib/umlmon/Name/jail/linux ist). Die Angabe dieses Parameters ist verpflichtend. Der Parameter darf in der dynamischen Konfiguration gesetzt werden.
- kernelarg: Ein Kernel-Argument. Dieser Parameter kann mehrfach gesetzt werden, um mehrere Kernel-Argumente zu übergeben. Der Parameter darf in der dynamischen Konfiguration gesetzt werden. Für User Mode Linux-Kernel sind ggf. folgende Argumente von Bedeutung:
  - devfs=nomount: schaltet die obsolete devfs-Funktionalität ab, falls sie versehentlich in den Kernel kompiliert wurde
  - hostfs=*Verzeichnis*: beschränkt die Möglichkeit, Dateibäume per HostFS zu mounten, auf das angegebene Verzeichnis und seine Unterverzeichnisse. Das Verzeichnis muss wie kernel relativ zum Jail-Verzeichnis angegeben werden (Verzeichnisse außerhalb des Jail-Verzeichnisses sind ohnehin nicht per HostFS zugänglich).
  - mode=tt: Erzwingt den TT-Modus, selbst wenn SKAS möglich ist
  - root=/dev/ubd*N*: Setzt die virtuelle Festplatte, die das Root-Image enthält. Wenn nicht angegeben, wird root=/dev/ubd0 benutzt.
  - single: Bootet im Single User-Modus
  - *N*: Bootet im Runlevel *N* (*N*=Zahl von 1 bis 5). Wenn nicht angegeben, wird der Runlevel aus /etc/inittab (des Gast-Systems) entnommen.
- mem: Die Größe des virtuellen Hauptspeichers in Megabytes. Die Angabe dieses Parameters ist verpflichtend. Der Parameter darf in der dynamischen Konfiguration gesetzt werden.
- mem_limit: Maximaler akzeptabler Wert von mem. Dieser Parameter darf nur in /etc/umlmon gesetzt werden.
- con*N*: Konfiguriert, dass die Konsole *N* von UMLMON gesteuert wird. *N* ist eine Zahl von 0 bis 63. Die Konsole 0 enthält normalerweise Boot-Meldungen. Die

Konsolen ab 1 können in `/etc/inittab` konfiguriert werden. Der Parameterwert von `conN` hat das Format

Methode`[:`*Option*`]`

Die einzige im Augenblick definierte Methode ist `pty`. Hierbei wird ein Pseudo-Terminal allokiert, über das die Ein- und Ausgaben geleitet werden. Als Option kann ein Dateiname angegeben werden, in der die Ausgaben geloggt werden. Die Datei wird im Log-Verzeichnis angelegt.

Beispiel: `con3=pty:con3.log` konfiguriert, dass die Konsole 3 von UMLMON zugänglich ist, und dass die Ausgaben in der Datei `con3.log` geloggt werden.

- `sslN`: Konfiguriert, dass die virtuelle serielle Schnittstelle *N* von UMLMON gesteuert wird. Der Parameterwert hat das gleiche Format wie `conN`.
- `ubdN`: Konfiguriert, welche Image-Datei den Inhalt der virtuellen Festplatte `ubd` *N* speichert. Der Parameterwert ist der Dateiname relativ zum Jail-Verzeichnis (wie bei `kernel`). Die Datei muss auf `.dsk` oder `.cow` enden und sie muss in einem der Verzeichnisse `/disks` oder `/shared/disks` existieren.

  Hinweis: Für COW-Dateien findet UMLMON automatisch die passende Backing-Datei. Diese muss daher nicht angegeben werden. Andererseits ist es nicht möglich, COW-Dateien vom Kernel erzeugen zu lassen.
- `ethN`: Konfiguriert das virtuelle Ethernet-Interface `ethN`. Folgende Werte sind für UMLMON zu empfehlen:
  - `host_to_host(...)`: Konfiguriert mit der Host-to-Host-Methode wie in Abschnitt 4.5 erläutert.
  - `bridged(...)`: Diese Syntax wird für die Einrichtung eines virtuellen Netzwerk-Segments benötigt (siehe Abschnitt 6.2).

  Im Prinzip sind jedoch alle Optionen möglich, die User Mode Linux unterstützt.

7.2 Das Kommando umladmin

Mit diesem Kommando können Monitore gestartet und gestoppt werden. Außerdem enthält das Kommando Hilfsfunktionen zur Konfiguration von UMLMON.

- `umladmin start`: Startet alle Monitore, die in `/etc/umlmon` konfiguriert sind, und für die `start=manual` oder `start=boot` gesetzt ist. Die Maschinen, für die `start=boot` gilt, werden außerdem hochgefahren.
- `umladmin start` *`Name1 Name2`* ...: Wenn Monitor-Namen übergeben werden, so werden genau diese Monitore gestartet und ggf. gebootet.
- `umladmin stop`: Fährt alle virtuellen Maschinen herunter und stoppt alle Monitore.

- `umladmin stop` *`Name1 Name2`* ...: Wenn Monitor-Namen übergeben werden, so werden genau diese Monitore gestoppt.

  `umladmin stop` hat eine Option:

  `-timeout` *`N`*: Wartet diese Zahl von Sekunden, bis die Shutdown-Skripte der Maschine mit dem Herunterfahren fertig sind. Nach *`N`* Sekunden wird die Maschine hart beendet. Wenn diese Option nicht übergeben wird, gilt *`N`*=300.

- `umladmin update`: Stoppt und startet Monitore, so dass wieder genau diejenigen Monitore aktiv sind, die in `/etc/umlmon` konfiguriert sind und für die `start=manual` oder `start=boot` gesetzt ist. Dieses Kommando kann nach dem Editieren von `/etc/umlmon` benutzt werden, um abzuschaltende Monitore zu stoppen und neue hinzugekommene Monitore zu starten. Wenn virtuelle Maschinen herunterzufahren sind, wird dies durchgeführt. Dieses Kommando fährt jedoch niemals Maschinen hoch.

  Wenn Monitor-Namen übergeben werden, wird der Update auf die genannten Monitore beschränkt.

  Die `-timeout`-Option gilt wie bei `umladmin stop`.

- `umladmin list`: Dieses Kommando gibt für alle konfigurierten Monitore aus, ob die Monitore aktiv sind und wenn ja, in welchem Zustand sie sich befinden.

  Optionen:

  `-active`: Nur solche Monitore werden gelistet, bei denen die virtuelle Maschine aktiv ist.

  `-inactive`: Nur solche Monitore werden gelistet, bei denen die virtuelle Maschine inaktiv ist.

  `-off`: Es werden nur Monitore gelistet, die konfiguriert aber ausgeschaltet sind

- `umladmin config` *`Name`*: Dieses Kommando gibt die tatsächliche Konfiguration der Maschine *`Name`* aus.

  Optionen:

  `-clear` *`param`*: Unterdrückt die Ausgabe dieses Parameters in der Ausgabe

  `-set` *`param=Wert`*: Setzt den Parameter auf den angegebenen Wert in der Ausgabe.

  `-rename` *`Name`*: Benennt die Maschine in der Ausgabe um

- `umladmin password`: Fragt nach einem Passwort und gibt den Hex-kodierten MD5-Hash des Passworts aus. Das Kommando kann interaktiv oder in Skripten benutzt werden.

7.3 Das Kommando umlwatch

Mit diesem Kommando kann eine virtuelle Maschine administriert werden, sofern der Monitor dieser Maschine gestartet worden ist. Es gibt einen Shell-Modus, in dem der Benutzer Kommandos interaktiv eintippen kann, sowie einen Skript-Modus, bei dem das (einzelne) Kommando schon auf der Kommandozeile übergeben wird.

Generell wird umlwatch aufgerufen durch:

```
umlwatch [Optionen] Name[@Host]
```

Wenn der Zusatz @Host angegeben wird, verbindet sich umlwatch per TCP-Verbindung mit dem Monitor auf dem Host.

Wenn keine Optionen angegeben sind, wird der Shell-Modus aufgerufen. Ansonsten erreicht man den Skript-Modus. Die umlwatch-Kommandos werden im Skript-Modus mit vorangestelltem Minus-Zeichen geschrieben. Beispielsweise wird das Shell-Kommando info zum Skript-Aufruf

```
umlwatch -info Name
```

Im folgenden wird das Minus-Zeichen jedoch weggelassen.

Liste der Kommandos:

- `info`: Gibt eine Übersicht der tatsächlichen Konfiguration der Maschine und ihres aktuellen Zustandes aus.
- `tail` *`Token`*`[,`*`N`*`]`: Gibt die letzten *`N`* Zeilen (oder 20 Zeilen, wenn nicht angegeben) der Log-Datei aus, die durch *`Token`* identifiziert wird. Das Token spricht die Datei unabhängig vom Namen an:
  - `monitor`: Die Log-Datei des Monitors wird ausgegeben
  - `con`*`N`*: Die Log-Datei der Konsole *`N`* wird ausgegeben
  - `ssl`*`N`*: Die Log-Datei der seriellen Schnittstelle *`N`* wird ausgegeben.
- `connect` *`Token`*: Verbindet das aktuelle Terminal mit dem virtuellen Terminal, das durch *`Token`* identifiziert wird, d.h. Eingaben des aktuellen Terminals werden als Eingabe zum virtuellen Terminal geleitet und Ausgaben des virtuellen Terminals werden im aktuellen Terminal dargestellt. Mögliche Token:
  - `con`*`N`*: Die virtuelle Konsole *`N`* wird verbunden.
  - `ssl`*`N`*: Die virtuelle serielle Schnittstelle *`N`* wird verbunden.

  Die Verbindung wird durch die Eingabe CTRL-] (d.h. Strg-AltGr-9 auf einer deutschen Tastatur) wieder gelöst. **Durch das Lösen der Verbindung erfolgt allerdings kein Log-Out!**

Verbindungen sind exklusiv, d.h. solange ein Benutzer mit einem virtuellen Terminal verbunden ist, kann sich kein anderer Benutzer mit dem selben Terminal verbinden. Inaktive Verbindungen (z.B. durch hängende Login-Prozesse) werden nach relativ kurzer Zeit automatisch gelöst.

- `force-connect` *`Token`*: Verbindet wie `connect`, aber mit dem Unterschied, das eine zuvor bestehende Verbindung zwangsweise gelöst wird.
- `start`: Fährt die virtuelle Maschine hoch
- `start-with` *`Arg`*: Fährt die virtuelle Maschine hoch und übergibt *`Arg`* als zusätzliches Kernel-Argument.

  Beispiel: `start-with single` fährt die Maschine im Single User-Modus hoch.
- `suspend`: Hält die Maschine vorübergehend an.
- `continue`: Bewirkt, dass die per `suspend` angehaltene Maschine fortgeführt wird.
- `halt`: Sendet einen HALT-Request an die virtuelle Maschine. Die Maschine beendet sich dadurch sofort, ohne Platten zu unmounten oder andere Aktionen durchzuführen.
- `ctrlaltdel`: Sendet einen CTRL-ALT-DEL-Request an die virtuelle Maschine. Die Maschine führt dadurch das in `/etc/inittab` konfigurierte Kommando aus. Dies ist üblicherweise das Kommando zum Herunterfahren der Maschine.
- `kill`: Beendet den Monitor-Prozess sofort. Da der Monitor-Prozess das „controlling terminal" des virtuellen Kernels ist, wird die Maschine dadurch sofort hart beendet.
- `sysrq` *`Buchstabe`*: Sendet einen SYSRQ-Request an die virtuelle Maschine. Der Buchstabe identifiziert den Request:
  - `b`: Die Maschine wird unmittelbar neu gestartet
  - `o`: Die Maschine wird unmittelbar beendet (wie halt)
  - `s`: Alle gemounteten Dateisysteme werden gesynct
  - `u`: Alle gemounteten Dateisysteme werden nur-lesend gemountet
  - `p`: Gibt die Prozessor-Register auf `con0` aus
  - `t`: Gibt alle Tasks auf `con0` aus
  - `m`: Gibt eine Beschreibung des Speicherverwendung auf `con0` aus
  - `e`: Sendet SIGTERM-Signale an alle Prozesse außer `init`
  - `i`: Sendet SIGKILL-Signale an alle Prozesse außer `init`

- `l`: Sendet SIGKILL-Signale an alle Prozesse inklusive `init`
- `0-9`: Die Ziffer setzt den Log-Level der Konsole.

- `rotate`: Öffnet alle Log-Dateien erneut. Dieses Kommando soll nach dem Rotieren der Log-Dateien (falls konfiguriert) aufgerufen werden.
- `wait` *N*: Wartet, bis die Maschine inaktiv ist, aber nicht länger als *N* Sekunden.
- `port`: Gibt den TCP-Port aus, auf dem der Monitor TCP-Verbindungen annimmt.
- `set` *Param*=*Wert*: Setzt den Parameter auf den angegebenen Wert in der dynamischen Konfiguration
- `del` *Param*: Löscht den Parameter in der dynamischen Konfiguration.
- `disk-create` *Name*:*N*[:sparse]: Erzeugt ein Image für eine virtuelle Festplatte in der Datei *Name*. Der Name muss die Form `/disks/`*Basisname*`.dsk` haben. Das Image hat *N* Megabytes Platz. Falls die Option `sparse` angegeben ist, wird der Platz nicht sofort auf der Platte allokiert, sondern erst dann, wenn er benötigt wird.
- `disk-create-cow` *Name*:*RName*: Erzeugt ein COW-Image für eine virtuelle Festplatte in der Datei *Name*. Das COW-Image benutzt *RName* als Referenz-Image. Der Name muss die Form `/disks/`*Basisname*`.cow` haben.
- `disk-delete` *Name*: Löscht das Image *Name*
- `disk-copy` *QName*:*ZName*: Kopiert das Image *QName* und speichert die Kopie unter *ZName*. Beide Namen müssen die gleiche Endung, „.dsk“ oder „.cow“, haben. *ZName* muss die Form `/disks/`*Basisname*`.`*Endung* haben.
- `disk-resize` *Name*:*N*[:sparse]: Ändert die Größe des Images Name auf *N* Megabytes ab. Nur `dsk`-Images, die nicht Referenz für COW-Images sind, können in der Größe geändert werden. Wenn die Option `sparse` angegeben ist, wird zusätzlicher Plattenplatz erst bei Bedarf allokiert.

  Wichtig: Dieses Kommando ändert die Größe des Images ohne den Inhalt des Images der neuen Größe anzupassen. Dies ist vergleichbar mit dem Ändern einer Partitionsgröße bei einem echten Rechner. Wenn der Inhalt ein Dateisystem ist, gilt folgendes:

  Bei Vergrößerungen des Images muss nach dem `disk-resize`-Kommando das Dateisystem entsprechend vergrößert werden, um den zusätzlichen Platz nutzen zu können.

  Bei Verkleinerungen des Images muss **vor** dem `disk-resize`-Kommando das Dateisystem bereits entsprechend verkleinert worden sein; **andernfalls droht Datenverlust!**

- `change-password`: Ändert das Passwort interaktiv ab.
- `site` *`arg`* `...`: Ruft das Site-Kommando mit den angegegeben Argumenten auf.
- `version`: Gibt die Version von umlwatch aus.
- `help`: Gibt einen Hilfetext aus.
- `exit`: Beendet die umlwatch-Shell.

A Übersetzung von passenden Linux-Kerneln

A.1 Stellt die Linux-Distribution passende Kernel zur Verfügung?

Da die Übersetzung von Linux-Kerneln nicht immer ganz einfach ist, sollte man zuerst prüfen, ob die Linux-Distribution, die man verwendet, geeignete Kernel zur Verfügung stellt.

Debian 3.1 (Codename Sarge): Es gibt z.Z. keinen UML-Gast-Kernel.

Ein SKAS3-fähiger Host-Kernel steht als Binary nicht zur Verfügung, kann aber durch Neuübersetzung mittels kernel-package relativ einfach erzeugt werden. Hierzu ist das Paket `kernel-patch-skas` zu installieren, und bei der Übersetzung muss der SKAS3-Patch angegeben werden. (Hinweise in Abschnitt C.1.1).

SuSE Linux 9.2: Die SUSE-Kernel enthalten bereits den SKAS3-Patch, so dass der Kernel nicht neu übersetzt werden muss. SUSE liefert auch Gast-Kernel mit, aber diese sind leider für den Einsatz unter UMLMON ungeeignet.

A.2 Host-Kernel 2.6 mit SKAS3-Unterstützung

Jeder Kernel 2.6 unterstützt bereits SKAS0. Wenn Sie jedoch maximale Performanz benötigen, können Sie einen Host-Kernel mit SKAS3-Features bauen.

Gehen Sie bitte wie folgt vor:

> **Hinweis:** Dies sind generische Anweisungen, die für jede Linux-Distribution gelten. Viele Linux-Distributionen haben jedoch einen eigenen Standard für die Kernel-Übersetzung entwickelt. Diese sind im Zweifelsfall geeigneter. In Anhang C haben wir für Sie Hinweise für einige Distributionen zusammengefasst.

1. Installieren Sie den Linux-Kernel-Quellcode, z.B. indem Sie einen Kernel von http://www.kernel.org herunterladen und das Archiv auspacken. Alternativ zu den Standard-Kerneln können Sie auch die Quellen verwenden, die Ihre Linux-Distribution mitliefert; diese sind häufig gepatcht und für die jeweilige Distribution optimiert. Die Anwendung des SKAS-Patches kann jedoch unter Umständen nicht funktionieren.

2. Laden Sie den passenden SKAS-Patch von http://www.user-mode-linux.org/~blaisorblade/patches/skas3-2.6/ herunter.
3. Wechseln Sie auf der Kommandozeile mit cd in das Linux-Verzeichnis, z.B.
   `cd .../linux-2.6.11`
4. Wenden Sie den SKAS-Patch an:
   `patch -p1 < `*`Verzeichnis`*`/skas3-2.6.11-v8.patch`
5. Konfigurieren Sie den Kernel. Normalerweise sollten Sie ihre alte Konfiguration weiter verwenden (meist im Verzeichnis `/boot` als Datei zu finden):
   `cp /boot/config-2.6.11 .config`

   Rufen Sie `make config` oder `make menuconfig` oder `make xconfig` auf, um die Konfiguration anzupassen. Stellen Sie sicher, dass folgende Optionen aktiviert sind (statisch oder als Modul):

   - In „Processor type and features“ aktivieren Sie „/proc/mm support“.
   - In „Device Drivers → Block devices“ sollte „Loopback device support“ aktiviert sein.
   - In „Device Drivers → Networking Support → Networking Options“ müssen „Unix Domain Sockets“, „TCP/IP Networking“ und „802.1d Ethernet Bridging“ aktiviert sein.
   - In „Device Drivers → Networking Support → Network Device Support“ muss „Universal TUN/TAP device driver support“ aktiviert sein.
6. Übersetzen Sie den Kernel:
   `make dep && make && make modules`
7. Installieren Sie den Kernel. Wie dies geschieht, hängt stark von Ihrer Distribution und dem Boot-Loader ab. Vergessen Sie nicht, auch die Module zu installieren.

A.3 Gast-Kernel 2.6

Um einen Gast-Kernel zu übersetzen, gehen Sie bitte wie folgt vor:

1. Installieren Sie den Linux-Kernel-Quellcode, z.B. indem Sie einen Kernel von http://www.kernel.org herunterladen und das Archiv auspacken. Alternativ zu den Standard-Kerneln können Sie auch die Quellen verwenden, die Ihre Linux-Distribution mitliefert; diese sind häufig gepatcht und für die jeweilige Distribution optimiert.

   Hinweis: Sie benötigen keinen Patch! User Mode Linux ist Teil des 2.6-Standard-Kernels.
2. Wechseln Sie auf der Kommandozeile mit cd in das Linux-Verzeichnis, z.B.
   `cd .../linux-2.6.11`

3. Konfigurieren Sie den Kernel:
   `make menuconfig ARCH=um`

   Hinweise: Rufen Sie in jedem Fall den Konfigurationsdialog auf und speichern Sie die Konfiguration, selbst wenn Sie nichts verändern. Der Zusatz `ARCH=um` ist **unbedingt notwendig**.

4. Übersetzen Sie den Kernel:
   `make dep ARCH=um && make linux ARCH=um`

5. Der übersetzte Kernel heißt `linux` und findet sich im Arbeitsverzeichnis. Sie sollten noch die Symboltabelle entfernen, bevor Sie ihn benutzen:
   `strip linux`

B Erzeugung von Root-Images

B.1 Welche Dateien müssen auf einem Root-Image zu finden sein?

Ein Root-Image ist ein Linux-Dateisystem, das das hochzufahrende System enthält. Es muss den Boot-Prozess ab Start von `/sbin/init` unterstützen. (Einen Boot-Loader, Boot-Sektoren oder ähnliches benötigen Sie auf dem Image nicht. Es muss sich auch kein Kernel auf dem Image befinden.)

Sie können ein solches Image grundsätzlich erzeugen, indem Sie ein auf einem echten Rechner ein Linux-System installieren, das so aussieht wie gewünscht, und die vorgefundenen Dateien in ein Image kopieren. Dies beschreiben wir näher im folgenden Abschnitt. Der Nachteil der Methode ist, dass Sie einen zusätzlichen Rechner benötigen. Für einige Linux-Distributionen beschreiben wir in Anhang C eine weitaus elegantere Methode: Das Image wird mit Hilfe von Installationsmedien weit gehend automatisch erzeugt. Sie benötigen zwar immer noch einen Rechner, auf dem Sie dies durchführen, und auf diesem muss bereits die gewünschte Distribution laufen. Dieser Rechner dient jedoch nicht als Vorlage, die kopiert wird, und kann daher beliebig konfiguriert sein.

In einem Image müssen die ubd-Gerätedateien eingerichtet sein. Sie erzeugen diese Dateien mit:

```
for n in 0 1 2 3 4 5 6 7; do
  mknod Ggf_Pfad/dev/ubd$n b 98 $((16*n))
done
```

Hinweis: Es ist auch eine andere Nummerierung mit ubda..ubdh gebräuchlich.

In `/etc/inittab` müssen Sie darauf achten, dass CTRL-ALT-DEL so konfiguriert ist, dass das System heruntergefahren und nicht neugestartet wird. Beispielsweise ändern Sie die Zeile

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

ab in (`-h` statt `-r`):

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now
```

Das `shutdown`-Kommando in `/etc/inittab` variiert je nach Linux-Distribution.

B.2 Root-Image von einem System abziehen

1. Loggen Sie sich auf dem System, das Sie in das Image kopieren wollen, als root ein.
2. Bestimmen Sie die Größe des Systems in Megabyte, inklusive des freien Platzes, der später benötigt wird.
3. Erzeugen Sie eine leere Image-Datei in der gewünschten Größe. Ersetzen Sie *N* durch die ermittelte Zahl an Megabytes:
   ```
   cd /
   dd if=/dev/zero of=root.dsk bs=1M count=1 seek=N
   mke2fs -j root.dsk
   ```
4. Mounten Sie das Image:
   ```
   mkdir root.dsk.fs
   mount -o loop root.dsk root.dsk.fs
   ```
5. Kopieren Sie die Dateien in das Image (erweitern Sie ggf. die Listen):
   ```
   cp -a /bin /dev /etc /lib /mnt /opt /root /sbin /usr /var \
    root.dsk.fs
   for d in tmp proc sys; do mkdir root.dsk.fs/$d; done
   ```
6. Erzeugen Sie die ubd-Gerätedateien:
   ```
   for n in 0 1 2 3 4 5 6 7; do
    mknod root.dsk.fs/dev/ubd$n b 98 $((16*n))
   done
   ```
   Editieren Sie `/etc/inittab` wie in Abschnitt B.1 beschrieben.
7. Falls Sie jetzt noch Modifikationen am Image vornehmen möchten, ist jetzt der richtige Zeitpunkt (z.B. Dienste abschalten, die vorläufig nicht benötigt werden etc.).
8. Abschließen:
   ```
   umount root.dsk.fs
   rmdir root.dsk.fs
   ```
   Die Datei `root.dsk` ist nun ein Image, das Sie zum Hochfahren eines User Mode Linux-Systems verwenden können.

C Hinweise für Linux-Distributionen

C.1 Debian

C.1.1 Einen Host-Kernel übersetzen

1. Installieren Sie die Pakete `kernel-package` sowie die Kernel-Quellen (`kernel-source-`*X* oder `linux-source-`*X*, wobei *X* die Versionsnummer ist)

   Hinweis: Benutzen Sie die mitgelieferten Kernel-Quellen und keine Standard-Kernel, da Debian einen Sonderweg bei initrd geht (cramfs).

2. Installieren Sie außerdem `kernel-patch-skas`.

3. Wechseln Sie nach `/usr/src` und packen Sie das Kernel-Archiv aus, z.B.:
   ```
   cd /usr/src
   tar xjf linux-source-2.6.8.1.tar.bz2
   ```

4. Kopieren Sie die aktuelle Kernel-Konfiguration:
   ```
   cp /boot/config-2.6.8.1-5-386 .config
   ```

5. Rufen Sie das Kommando auf:
   ```
   make-kpkg --added-patches skas --append-to-version skas3 \
    --config menuconfig --initrd kernel_headers kernel_image
   ```

   Ihnen wird nun eine Frage bezüglich initrd gestellt: „Warning: You are using the initrd option ... Should I abort[Ny]?“ Drücken Sie einfach auf Enter.

   Daraufhin kommen Sie in den Kernel-Konfigurationsdialog. Sie haben nun die Möglichkeit, Änderungen an der Konfiguration vorzunehmen. Es ist aber in Ordnung, nichts zu ändern, und sofort mit „Exit“ den Dialog zu verlassen. Speichern Sie die Konfiguration.

6. Im übergeordneten Verzeichnis finden Sie nun zwei Debian-Pakete `kernel-headers-`*X* und `kernel-image-`*X*, die Sie installieren können:
   ```
   dpkg -i kernel-headers-...deb
   dpkg -i kernel-image-...deb
   ```

C.1.2 Erzeugung eines Root-Images

1. Installieren Sie das Paket `debootstrap`.

2. Führen Sie folgendes Kommando aus:
   ```
   debootstrap sarge root http://ftp.freenet.de/debian
   ```

   Dies erzeugt ein Unterverzeichnis namens „root“, in dem die Sarge-Distribution installiert ist.

   Hinweis: Das `debootstrap`-Paket muss von einem aktuellen Sarge-System stammen, da in ihm das Wissen einprogrammiert ist, welche Software und z.T. welche Software-Versionen zu einem Sarge-System gehören. Die HTTP-URL können Sie durch einen beliebigen anderen Debian-Mirror ersetzen.

3. Erzeugen Sie die ubd-Devices:
```
for n in 0 1 2 3 4 5 6 7; do
  mknod root/dev/ubd$n b 98 $((16*n))
done
```
   Editieren Sie `/etc/inittab` wie in Abschnitt B.1 beschrieben.

4. Löschen Sie Dateien, die nicht mehr gebraucht werden:
```
rm -rf root/var/cache/apt/archives/*
```

5. Erzeugen Sie die Image-Datei:
```
dd if=/dev/zero of=root.dsk bs=1M count=1 seek=500
mke2fs -j root.dsk
mkdir root.dsk.fs
mount -o loop root.dsk root.dsk.fs
cp -a root/* root.dsk.fs
umount root.dsk.fs
rmdir root.dsk.fs
```
   Die Datei `root.dsk` kann nun als Root-Image benutzt werden.

   Hinweise: Es ist kein root-Passwort gesetzt. Das System ist völlig unkonfiguriert und beinhaltet nur die notwendigste Software. Sie kommen aber von hier aus weiter: Starten Sie das System, konfigurieren Sie ein Netzwerk, tragen Sie in `/etc/apt/sources.list` eine Software-Quelle ein und installieren Sie mit `apt` oder `dselect` weitere Pakete. Es ist vorteilhaft, wenn Sie `root.dsk` so groß wie hierfür nötig machen; ersetzen Sie die Zahl 500 in obigen Kommandos durch die Zahl Megabytes, die Sie für ihr System erwarten.

   Debian liefert ein weiteres Tool namens `rootstrap` mit, das zur Erzeugung von Root-Images vorgesehen ist. Bei Tests funktionierte dieses Tool jedoch nicht richtig.

C.2 SUSE

SUSE liefert einen Gast-Kernel (im Paket `um-host-kernel`). Dieser ist leider dynamisch gelinkt (sogar gegen `libX11`!) und daher für den Einsatz unter UMLMON ungeeignet.

Im Paket `uml-utilities` gibt es ein Skript `uml-install-suse`, mit dessen Hilfe man ein Root-Image erzeugen können soll. Wir haben das mit diesem Skript allerdings nicht geschafft, es ist offensichtlich fehlerhaft.